vocabotics

A VOCABOTICS AGENT GUIDE

THE AGENT GUIDE

The Beginner's Guide to AI Agents

What an AI 'agent' actually is — not a chatbot, not quite a robot — one real agent doing a real job shown step by step including the moment it gets something wrong, an honest vendor scorecard, and the stop-button mindset that lets you use the upside without betting anything on it.

For anyone deciding whether to trust an agent with something real

Shipped — measured, gate-clean

COVER: AN EMERGENCY STOP BUTTON, CLOSE UP — REAL PHOTOGRAPH, BISO (WIKIMEDIA COMMONS, CC BY-SA 4.0).

vocabotics

A VOCABOTICS AGENT GUIDE

The Beginner's Guide to AI Agents

What an AI 'agent' actually is — not a chatbot, not quite a robot — one real agent doing a real job shown step by step including the moment it gets something wrong, an honest vendor scorecard, and the stop-button mindset that lets you use the upside without betting anything on it.

PREPARED
5 July 2026

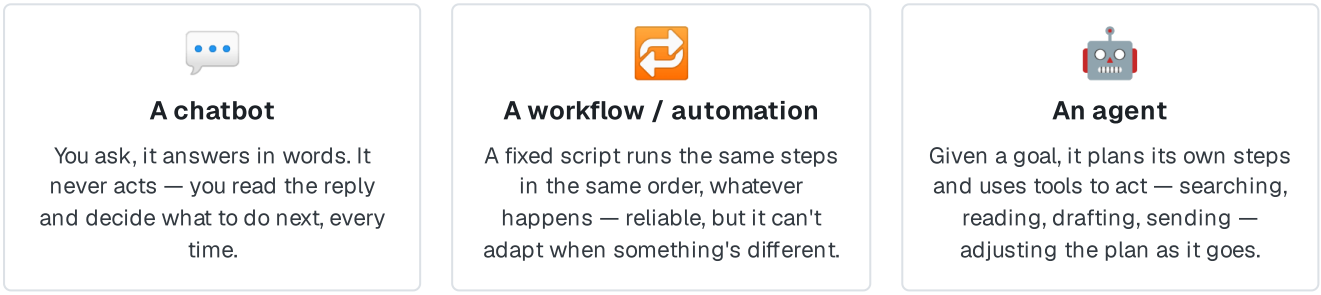
VERIFIED BY
Jon Ross — Founder, vocabotics — 15 years building safety-critical AI

HOW TO READ IT
Drafted with AI, verified by a named human. Sources at the back.

01 What an AI agent actually is

Three words get used almost interchangeably — chatbot, workflow, agent — and they are not the same thing. Here is the honest difference.

"AI agent" is the phrase every vendor is currently using, on almost every product, to mean almost anything. This guide exists because that vagueness has a real cost: it makes it hard to tell a genuinely useful new capability from a rebadged chatbot, and it makes it easy to buy — or build — something with more autonomy than you actually want to hand out. So before anything else, three plain definitions.



Three things that get called 'AI', and what actually separates them.

The distinction that matters is **who decides the next step**. A chatbot hands that decision to you, every single time — nothing happens until you read the reply and act on it yourself. A workflow decides it in advance, once, when someone wrote the automation — step 2 always follows step 1, and if step 1's result is slightly unusual, the workflow doesn't notice. An agent decides it *live*, case by case, based on what just happened — which is exactly what makes it more useful on a messy real-world task, and exactly what makes it riskier to leave alone.

	ADAPTS WHEN SOMETHING'S UNUSUAL	HANDLES A MULTI-STEP REAL TASK	PREDICTABLE, SAME RESULT EVERY TIME	SAFE TO LEAVE FULLY UNATTENDED
A chatbot	◐	○	–	●
A fixed workflow	○	◐	●	◐
An agent	●	●	○	○

● Go ◐ Check it ○ Keep a human

The same three approaches, scored on the four things that actually matter for trusting them with something real.

Read the last two columns carefully — they're the whole point of this guide. An agent is the most capable of the three exactly because it is the least predictable and the least safe to leave unattended. That trade isn't a

flaw to be engineered away by next year's model; it's the honest shape of the thing. Everything from here on is about getting the capability without eating the risk.

THE PLAIN TRUTH **The advisor vs. the assistant with your keys**

Think of a chatbot as an advisor: you ask, it tells you, you decide and act. Think of an agent as an assistant you've handed a set of keys: it can open doors and do things on your behalf, without checking every single one back with you first. Both can be useful. Only one of them needs a stop button.

02 What an agent is not

This is where the hype does real damage — so it's worth being blunt.

None of what follows is a hedge or a legal disclaimer bolted on afterwards. It's the honest boundary of what today's agents are, stated plainly because a guide that hedges on this is worse than no guide at all.

THE PITCH	THE HONEST REALITY
"It's like hiring a reliable employee."	It has no judgement, no accountability and no common sense about your business. It follows patterns; it doesn't understand stakes the way a person who could lose their job does.
"It understands what it's doing."	It predicts a plausible next step. It can be confidently, fluently wrong while sounding exactly as certain as when it's right — there is no tone-of-voice tell.
"Set it loose and it'll sort itself out."	The more freedom and access it has, the more damage a single wrong step can do — and wrong steps happen, at every model generation, including this one.
"It's basically a small robot brain."	It's software that calls other software. There is no inner life to appeal to and no instinct for self-preservation that keeps it cautious — only the limits you build in from outside.

What the pitch says, and what's actually true.

The limits that already applied to an ordinary chatbot — making facts up, not truly understanding a situation, occasionally misreading what you asked for — do not disappear when you give the same model the power to act. They get **more consequential**, because a mistake is no longer a bad sentence you can shrug off. It's an email actually sent, a booking actually made, a file actually changed.

> ***A chatbot's mistake is a bad paragraph. An agent's mistake is a wrong email that already went out.***
>
> vocabotics — the whole reason this guide has a control chapter

03 The anatomy of an agent

Six parts, in order — the shape every agent shares, underneath the branding.

Strip away any specific vendor's marketing and every agent — a coding assistant, a browser agent, an inbox drafter — is built from the same six parts, wired into a loop. Understanding this anatomy is what turns "agent" from a buzzword into something you can actually reason about, ask questions of, and put guardrails on.

1 **Goal** what you asked for	**2** **Plan** the steps it thinks it needs	**3** **Tools** what it can actually touch
4 **Memory** what it keeps from earlier steps	**5** **Loop** act, observe, re-plan, repeat	**6** **Stop** where a human must say yes

The six parts every agent is built from.

1–2. Goal and plan

You give it a **goal** in plain language — "find three suppliers of X near Leeds and draft an intro email to each." The model then writes itself a **plan**: a rough sequence of steps it believes will get there. This plan is not fixed in advance by a programmer, the way a workflow's steps are — it's generated fresh, for this goal, and it can be wrong. A vague goal tends to produce a vague, over-ambitious plan; a specific goal with clear boundaries produces a specific, checkable one. The single highest-leverage thing you control is the goal you write.

3. Tools

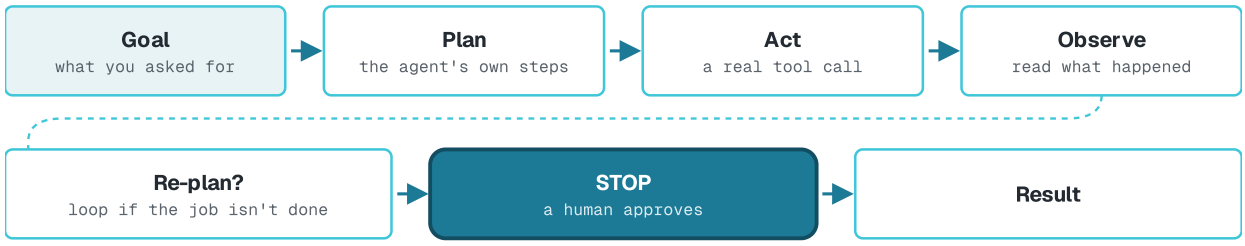
A **tool** is anything the agent can call to affect the world outside its own reply — a web search, a file read, an email-send function, a database query, a button-click in a browser. This is the entire difference from a chatbot in one sentence: a chatbot's only "tool" is writing text back to you; an agent has a toolbox of real actions, each with real consequences and a real cost of being wrong. Anthropic and other labs have converged on a shared, open way for a model to discover and call tools consistently (the Model Context Protocol is one such standard) — worth knowing the name of, because "which tools can it use, and who decided that" is the single most important question in this whole guide.

4–5. Memory and the loop

Memory is what the agent carries forward between steps — the search results it already read, the draft it already wrote, the fact that step 2 failed. Then it **loops**: act, observe what happened, decide whether the plan still holds or needs revising, act again. This loop is where the useful adaptability lives — and it's also where a small early error can compound, because each step's "observe" is built on the previous step's possibly-wrong output.

6. Stop

The **stop** is not a feature an agent has by default — it is a boundary you (or the tool you're using) have to deliberately build in: a point where the loop pauses and a human must say yes before anything irreversible happens. Everything from Chapter 8 onward is really about this one part, because it's the part that decides whether an agent's mistake costs you a minute of re-reading or a customer.



The loop, drawn in order — the stop-gate is the part that has to be built in.

Where vocabotics has actually built this

Our own lab has run this exact anatomy in production — a standard "agent contract" so every part of a system, down to the gateway routing work between agents, is itself an agent with the same shape. It's *research-tier* work, not a shipped consumer product, and it comes with its own honest limits (agents write missing skills for small increments well; they don't make the big architectural calls). We mention it because the anatomy above isn't theoretical — it's the same six parts we build with. The full write-up is public: see *Everything is an agent* in our Lab Notebook.

A tool is anything an agent can actually touch — here, the physical control panel that turns a plan into a real action.

Source: Photo: a robotic arm's control panel in a factory — Shixart1985, Wikimedia Commons, CC BY 2.0

04 A real agent run, step by step — including where it goes wrong

This is the chapter the old guide skipped. One goal, one agent, every step shown — the plan, each tool call, the mistake, and the human who caught it.

Everything above is abstract until you watch it happen. Here is a single, realistic run: a small business owner asks a bounded research-and-drafting agent to find three suppliers and draft intro emails. The task, the tool calls, the plan and the mistake below are illustrative — built to be exactly the shape of what really happens with today's agents, not a transcript of one specific product — but nothing here is softened for effect. This is what a genuinely well-run agent session looks like, warts included.

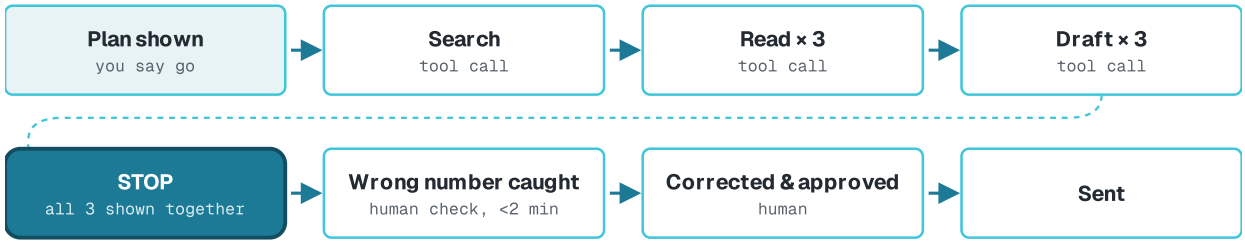
WORKED EXAMPLE

The goal: "Find three suppliers of food-grade packaging tape near Leeds, and draft a short intro email to each."

1 · THE PLAN	The agent replies with its intended steps before doing anything: search for suppliers → open the most promising three results → extract a name, website and phone number for each → draft a short intro email per supplier → stop and show you everything before sending. You skim it — it's sensible, bounded, and nothing irreversible happens without a further step. You say go.
2 · TOOL CALL — WEB SEARCH	`search("food grade packaging tape supplier Leeds") → 9 results returned, ranked by the search tool, not verified by the agent.`
3 · TOOL CALL — READ PAGE 1	`open("northernpackagingsupplies.co.uk/tape") → page text extracted: company name, a "call us" banner, a products list. The agent notes the name and the number shown in the banner.`
4 · TOOL CALL — READ PAGES 2 AND 3	`Same pattern repeated for two more supplier sites. All three now have a name, a website and a phone number recorded in the agent's working memory for this run.`
5 · THE PLAN UPDATES ITSELF	Nothing unusual happened, so the agent doesn't deviate — it moves straight to drafting, exactly as step 1 said it would. This is the loop working as intended: no news, no re-plan.
6 · TOOL CALL — DRAFT THREE EMAILS	Three short, polite intro emails are drafted, each personalised with the supplier's name and a line about food-grade tape specifically. None have been sent — the agent's plan explicitly routes to a stop before that tool is used.

7 · STOP — THE APPROVAL GATE YOU	The run pauses. All three drafts are shown together with the extracted supplier details, and nothing further happens until a person reviews them.
8 · THE NEAR-MISS, CAUGHT YOU	Reading supplier two's draft against its actual website in a separate tab, the phone number doesn't match — the agent had pulled a number from an ad banner further down the same page, not the company's real contact number in the footer. A plausible-looking number, attached with total confidence, that was simply the wrong one on the page.
9 · THE CORRECTION YOU	The reviewer corrects the phone number by hand, checks the other two against their sites (both correct), and approves all three to send. The whole check took under two minutes.
10 · SENT	The three approved emails go out. The run ends. Nothing was undone.

Nothing about this run was exotic. The agent did exactly what it said it would, saved a genuine twenty minutes of manual searching, and still confidently attached a wrong phone number to a real business — not because it was a bad model, but because a web page had two numbers on it and it picked the wrong one with the same fluent certainty it used for the right two. The stop-gate is the only reason that error cost two minutes of checking instead of a supplier receiving a slightly odd first contact. Remove step 7 and this run reads identically right up until the point a wrong number goes out attached to your company's name.



The same run, as a timeline — five real tool calls, one deviation-free loop, one human gate that caught the only mistake.

05 The same task, two setups — safe and reckless

Identical goal, identical model. Only the permissions and the gate change — and that's the entire difference in outcome.

The run above didn't go well because the model is unusually careful. It went well because of two decisions made *before* the agent ever started: it could only read the web and draft, not send; and a human had to look at all three drafts together before anything left the building. Change only those two things, keep everything else identical, and here is the same wrong phone number in a different setup.

Same goal, same model, same mistake at step 3 — different setup, completely different outcome.

Bounded & supervised

- Least access: can read the web, cannot send anything
- Every draft shown together before it leaves
- Wrong number caught in under two minutes, cost: nothing
- One clear point where a person is accountable

VS

Broad access, no gate

- Given full inbox-send access 'to save a step'
- No pause — drafts route straight to send
- Wrong number goes out under your company's name
- First anyone hears of it is the supplier's reply

What actually differs, side by side.

The lesson is not 'agents are risky'

The lesson is narrower and more useful than that: **the model made the same mistake in both setups.** The only thing that changed the outcome was whether a human saw the output before it became irreversible. That is the entire game, and it's why the next two chapters are the ones worth actually acting on.

06 What's reliable today — and what's still hype

Candour matters more here than anywhere else in this guide.

The honest picture, as of mid-2026

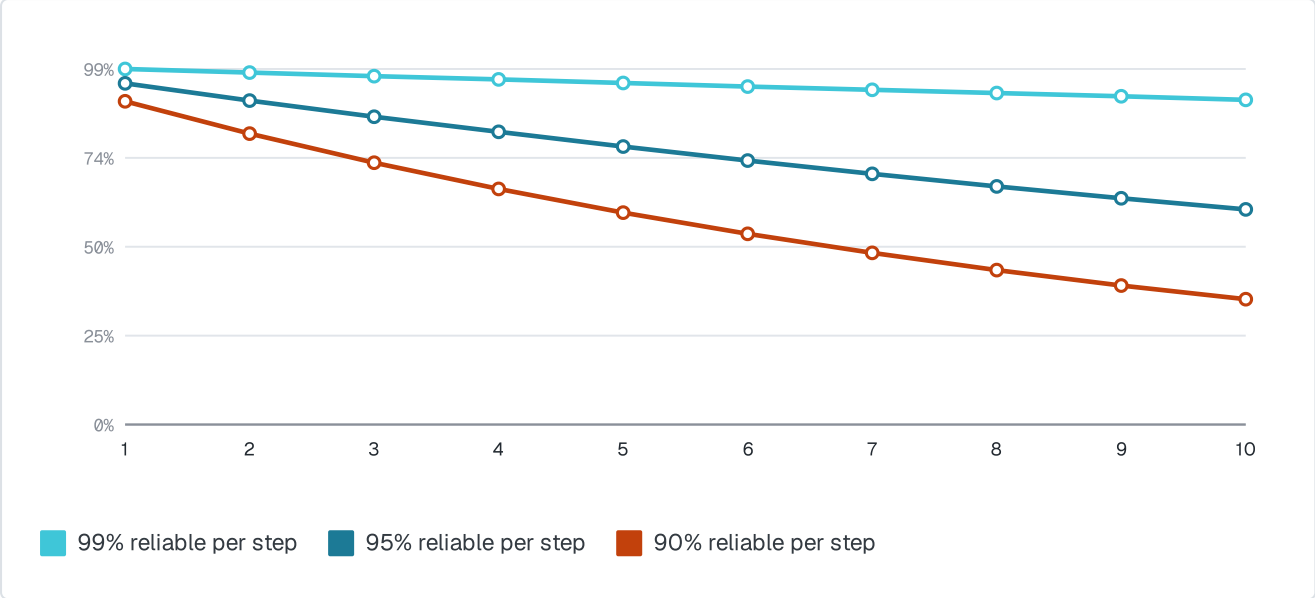
Our AI Reliability Traffic Light, applied to agents specifically. Green means it genuinely works today; amber means it's improving fast but still needs a human check; red means treat any claim otherwise as marketing, not fact.

A narrow, bounded task with a human approving each real action Works well today — drafting for you to send, or gathering information	Go as of July 2026
Reading and summarising documents or web pages as part of a plan Strong — the same long-document strength as an ordinary assistant, chained	Go as of July 2026
Chaining many steps unattended for several minutes Improving fast, but a wrong step early derails everything after it	Amber as of July 2026
Knowing when it's actually finished vs. merely out of ideas Agents can loop past the sensible stopping point without noticing	Amber as of July 2026
Fully autonomous agents running your business unsupervised Not reliable enough today; treat any such claim as marketing, not fact	Stop as of July 2026

The single most valuable sentence in this whole guide is about arithmetic, not opinion. If one step in a chain is 90% reliable — a genuinely good number for an individual step today — ten of those steps chained together are not 90% reliable overall. They compound.

$$0.9^{10} \approx 0.349$$

Ten steps at 90% reliability each: 0.9 multiplied by itself ten times is about 35%, not 90%.



The corpus's most important sentence, made into a picture: reliability per step vs. reliability over a chain of ten.

Read the bottom line first: even at a genuinely good 90% per-step reliability, a ten-step unattended chain is right about a third of the time overall — worse than a coin flip on getting through cleanly. This is not a criticism of any particular model; it's arithmetic, and it holds regardless of which vendor's agent you're looking at. It is also exactly why the honest advice in this guide is never "let it run for longer," but always "put a gate somewhere in the middle of the chain, not just at the end."

07 The agents you've actually heard of, sorted honestly

Coding agents, browser agents, inbox agents — the same go/amber/stop logic, applied to what's actually on the market.

"Agent" isn't one product category — by mid-2026 it's a handful of quite different tools wearing the same word. Here's the honest sort, by what each is actually asked to do and what happens when it's wrong.

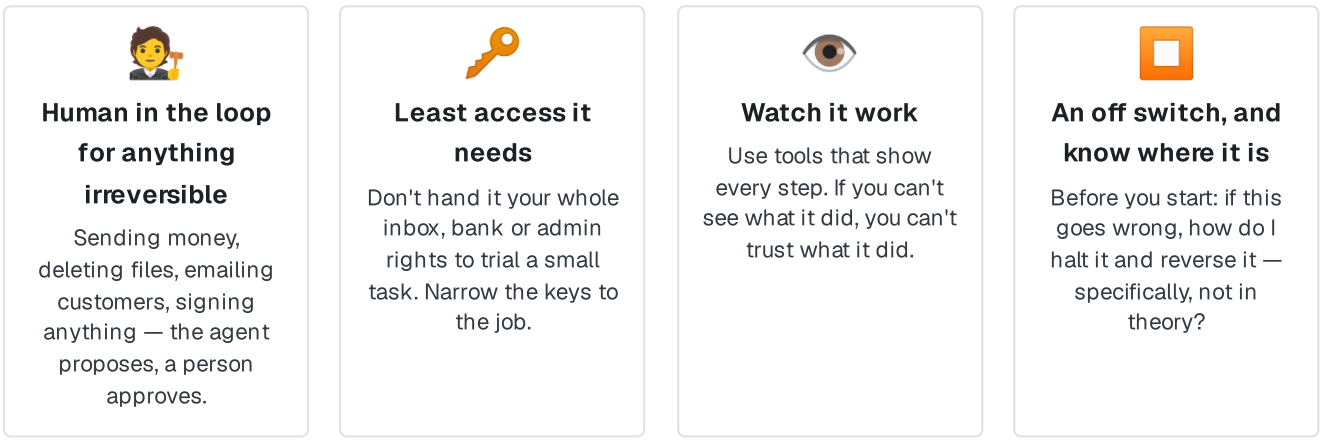
A coding agent making a small, scoped change you review before it merges A wrong line is caught in review; nothing ships without a human's approval	Go	as of July 2026
A browser agent filling in a web form while you watch each field You catch a wrong field before submission, every time	Go	as of July 2026
An inbox agent drafting replies for you to read, edit and send yourself All the time saved, none of the 'it emailed someone something wrong' risk	Go	as of July 2026
A coding agent given commit-and-deploy access with no review step The same wrong line now ships to production before anyone reads it	Stop	as of July 2026
A browser agent given saved payment details to 'just handle checkout' A wrong item, quantity or address becomes a real, often hard-to-reverse charge	Stop	as of July 2026
An inbox agent set to auto-send replies with no human reading them first The exact failure from Chapter 5's second setup — nothing catches the error	Stop	as of July 2026

Notice the pattern repeats across every category: it is never the underlying capability that flips a row from green to red. It's always the same lever — whether a human sees the output before it becomes irreversible. Learn that one lever and you can sort any new "agent" product that launches after this guide is printed, without needing a new chapter for it.

08 The control chapter — guardrails, not vibes

Everything about using agents safely comes down to one principle, and four practical guardrails under it.

A human must always be able to see what an agent is doing and stop it. That's the whole principle. National safety and cyber-security bodies frame it the same way, in almost the same words: design for control, not blind autonomy — keep humans in charge, limit what a system can touch, and monitor it while it runs.



The four guardrails you can hold in your head, in practice.

What a real approval gate actually looks like

"Human in the loop" is easy to say and vague to picture, so here's what it looks like in practice, in the tools worth using: a screen that shows the agent's proposed next action *before* it happens, in plain language, with an explicit approve/reject control — not a settings toggle you switch on once and forget. A genuine audit log lets you scroll back through every step a run took, in order, after the fact — not a vague "activity summary," but the actual sequence of tool calls, timestamped. If a product can't show you that log, you are being asked to trust it on faith, which is precisely the thing this guide argues against.

Cost and loops — the guardrail nobody markets

An agent that gets stuck doesn't fail quietly — it tends to keep trying. A step that doesn't complete cleanly can trigger another attempt, which triggers another, and every attempt is a real, metered cost: more model calls, more tool calls, more time. A runaway loop rarely announces itself; it just quietly runs up a bill or hammers a rate limit until something else breaks. Set a hard cap before you start — a maximum number of steps, a maximum spend, a maximum run time — and treat hitting that cap as information (something about the plan is wrong), not an obstacle to raise the cap past.

The rule to tattoo on the back of your hand

Give an agent power in proportion to how easily you can undo its mistakes. Reversible and low-stakes? Let it run. Irreversible or costly? A human approves every action, every time — no exceptions for "just this once, it's a small one."

Watching it work, in real time — the guardrail that costs the least and catches the most.

Source: Photo: a telescope operator at work in the VLT control room, Paranal – ESO, Wikimedia Commons, CC BY 4.0

09 ◎ The vendor scorecard — score any agent pitch against this

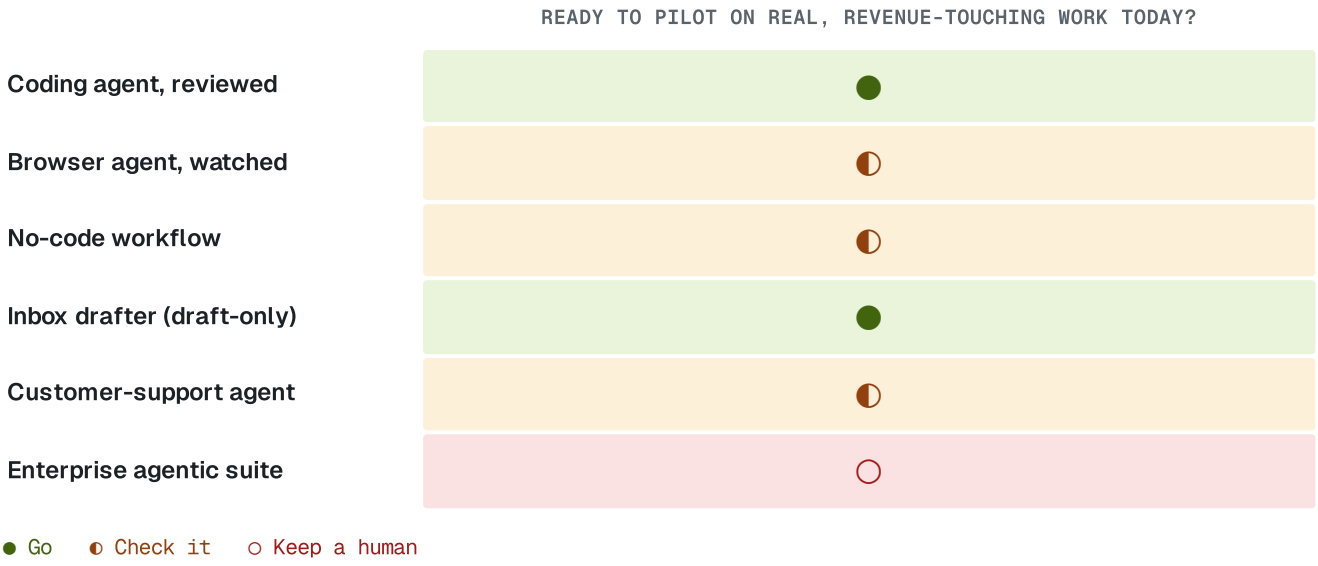
Six approaches on the market today, scored honestly on the things that actually decide whether they're safe to pilot.

This is the instrument the old guide's "questions to ask a vendor" list was missing — a real scorecard, not just questions. Score any product pitched to you (0 = absent, 1 = partial or bolted-on, 2 = solid, 3 = built-in and hard to disable) against the same six rows, and you have a comparable, defensible number instead of a gut feeling.

APPROACH	SEE EVERY STEP?	LIMIT ACCESS?	APPROVAL GATE?	STOP & UNDO?	AUDIT LOG?	BEST FIT TODAY
Coding agent (IDE / terminal)	3	2	3 (diff review)	3 (git revert)	3	Scoped changes you review before merge
Browser / "computer-use" agent	2	1	1 (varies a lot)	1	1	Watched, single-session tasks only
No-code workflow automation	2	2	1 (rare by default)	2	2	Repetitive, low-stakes, well-tested flows
Inbox / email drafting agent	2	2	2 (draft-only is the safe mode)	2	1	Drafts for you to send yourself
Customer-support / helpdesk agent	1	1	1 (escalation rules only)	1	2	FAQ triage, handed off before anything binding
Enterprise "agentic platform" suite	1	1	varies (ask, don't assume)	1	2	Pilot on one narrow process, never the whole org

The six approaches on the market in mid-2026, scored 0–3 on each guardrail. Fill in your own vendor's row and compare.

Our own honest, dated scoring — a snapshot to recheck, not an independent lab benchmark. Score the specific product in front of you; categories vary a lot between vendors.



The same six approaches, at a glance — ready to pilot on real work today?

Don't buy below this line

Add the five numeric columns for a total out of 15. We would not pilot anything scoring below 6 overall, and would never proceed — regardless of the total — if "Approval gate" or "Stop & undo" individually score 0. Those two rows are not nice-to-haves; they are the entire difference between Chapter 5's two setups.

Questions to ask before you trust any "agent" product

- [ ] What exactly can it do — what actions, on what systems?
- [ ] What can it access, and can I limit that myself?
- [ ] Where does a human have to approve, and can I add approval steps it doesn't ship with?
- [ ] Can I see every step it took, afterwards, in order?
- [ ] How do I stop it mid-run, and undo what it already did?
- [ ] Is there a hard cap on cost, steps or run time — and what happens when it's hit?

10 Is this a job for an agent? — the decision

A straight, honest checklist — not every task wants this tool.

The most common mistake we see is not choosing an unsafe agent — it's pointing a genuinely good agent at the wrong kind of task. Agents earn their keep on multi-step, tool-using, somewhat-tolerant-

of-a-retry jobs. They are the wrong tool for anything where a single wrong step is expensive, irreversible, or hard to check afterwards — for those, a chatbot you read every reply from, or a fixed workflow you've tested, is simply the better instrument.

Does the task need several real steps (search, read, act), not just one written answer?	**Yes** → An agent is worth considering. Continue.	**No** → Use a plain chatbot — you'll get the same result faster and safer.
Is every step it would take reversible, or cheap to check afterwards?	**Yes** → Good sign. Continue.	**No** → Add a hard approval gate before anything irreversible — never skip Chapter 8.
Does this exact sequence run the same way, unchanged, every single time?	**Yes** → A fixed workflow is more predictable and usually cheaper — use that instead.	**No** → An agent's ability to adapt the plan is genuinely earning its place here.
Can you name, specifically, how you'd undo a mistake in the next ten minutes?	**Yes** → You're ready to pilot it — see Chapter 12's two-week plan.	**No** → Stop. Design the undo path first; an agent without one is a liability, not a shortcut.

Five minutes to decide, honestly.

AGENT-SHAPED	NOT AGENT-SHAPED
"Find three suppliers and draft intro emails" — multi-step, reversible, checkable	"Send this month's payroll" — one step, irreversible, expensive if wrong
"Tidy and reformat a copy of this spreadsheet" — the original is untouched either way	"Reconcile the live accounts" — a wrong figure propagates before anyone reads it
"Research and summarise, for me to act on" — you stay the decision-maker	"Decide and act on my behalf" — nobody's reading it before it happens

The pattern, stated plainly.

11 A safe first agent to actually try

Not a description of one. A recipe — fill in your own details and run it.

Don't start by automating your business. Start with something bounded, useful, and genuinely easy to check — the same shape as Chapter 4's worked run, scaled to whatever you actually need done this week.

WORKED EXAMPLE

TEMPLATE — your first safe agent task

1 · PICK A RESEARCH OR DRAFTING JOB, NEVER A SENDING JOB	"Find three [suppliers / venues / candidates] matching [your criteria] and give me the website, phone number and a one-line summary for each" is the archetype. It gathers; you verify and act.
2 · SET THE TOOLS TO READ-ONLY	Web search and page-reading only. No send, no purchase, no file-delete access — even if the tool offers it. You add access later, deliberately, once you trust the pattern.
3 · ASK FOR THE PLAN BEFORE IT STARTS	`"Show me your planned steps before you take any action." A sensible plan is short, bounded, and ends in showing you the result — not in sending or spending anything.`
4 · SET A HARD CAP	A maximum number of steps or a time limit, agreed before you start. Hitting the cap means stop and reassess, not "let it keep going a bit longer."
5 · READ EVERY OUTPUT AGAINST ITS SOURCE YOU	Exactly as in Chapter 4: check the extracted detail against the real page it came from, not just against how confident the summary sounds.
6 · ONLY THEN, CONSIDER MORE ACCESS YOU	Once you've run the bounded version a few times and trust the pattern, consider adding a narrow, specific extra permission — never "just give it everything to save a step."

This recipe costs you nothing but ten minutes and produces a genuinely useful result on its first run — and it teaches you the shape of a safe agent session before you ever risk a real one.

A bounded, collaborative robot working directly alongside a person — the physical version of 'least access, human right there.'

Source: Photo: Rodney Brooks with Baxter, a collaborative robot designed to work safely alongside people — Steve Jurvetson, Wikimedia Commons, CC BY 2.0

12 Template: the guardrail pre-flight checklist

Run this before you switch any agent on for real — lift it, keep it, use it every time.

Before you press go

- [ ] The goal is specific enough that a sensible plan is short and checkable.
- [ ] Tools are set to the least access the task needs — no blanket admin, inbox or payment access.
- [ ] There is an explicit approval gate before anything irreversible happens.
- [ ] You can see every step it took, in order, after the fact.
- [ ] You know exactly how to stop it mid-run and undo what it's already done.
- [ ] A hard cap is set on steps, cost or run time, agreed before you start.
- [ ] Someone specific — named, not 'the team' — is accountable for reading the output before it goes anywhere consequential.

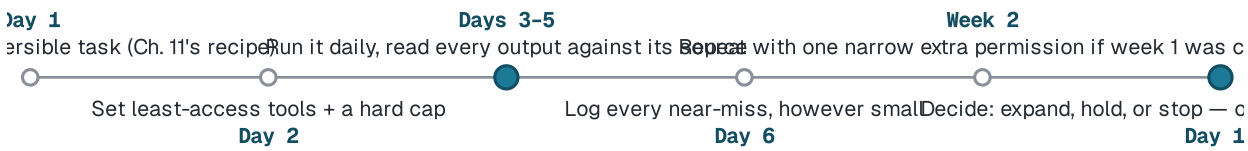
KEEP A HUMAN HERE **If you can't tick all seven**

Don't run it on anything real yet. Every item on this list is cheap to fix before you start and expensive to discover you needed after something's already gone out.

13 Piloting one safely — a two-week plan

The 'Monday' plan — what to actually do, day by day.

This is the plan we'd actually run, on a single narrow task, before trusting an agent with anything wider.



Two weeks, start to a real decision.

SIGNAL	GREEN — EXPAND A LITTLE	RED — HOLD OR STOP
Near-misses	None, or caught trivially at the gate every time	Any near-miss that would have been costly without the gate
Time spent checking	Minutes per run, and falling	Checking takes nearly as long as doing it yourself
Cost / step count	Stable, within the cap, no loops	Regularly hitting the cap or trending up

What 'clean' actually means, so day 14's decision isn't a guess.

14 The honest limits — agents fail, and that's the whole point of this guide

Not a caveat at the end. The reason every chapter before this one exists.

Every guardrail in this guide exists because agents genuinely fail — not occasionally, in edge cases, but routinely, in ordinary ways. They hallucinate plans that sound sensible and aren't. They loop past the point a person would have stopped. They cost real money on a bad run. And sometimes the failure is simply, delightfully dumb.

A real, documented failure: autonomous delivery robots that got close enough to each other that none of them could move — each one's own collision-avoidance rule left it stuck.

Source: Photo: Autonomous Delivery Vehicle Pileup – Mbrickn, Wikimedia Commons, CC BY-SA 4.0

Nobody designed that pileup. Each robot was independently following a reasonable rule — don't get too close to another robot — and the combination of several reasonable individual rules produced a collectively useless outcome that no single robot's logic would predict from the inside. That is the honest shape of agent failure in general: not a moment of obvious malice or an incomprehensible glitch, but small, locally sensible decisions compounding into a result nobody wanted and nobody would have chosen on purpose.

FAILURE	WHAT IT LOOKS LIKE	THE GUARDRAIL THAT CATCHES IT
Hallucinated plan	A confident, sensible-sounding step that doesn't actually achieve the goal	Reviewing the plan before it starts (Ch. 4, step 1)
Confident wrong detail	A fact, number or contact detail stated with the same certainty as a correct one	Checking output against its source (Ch. 4, step 8)
Runaway loop	Repeated retries that don't converge, quietly burning time and money	A hard cap on steps, cost or run time (Ch. 8)
Compounding drift	A small early error that later steps build on without re-checking	A gate partway through a long chain, not just at the end (Ch. 6)
Emergent gridlock	Individually sensible rules combining into a useless collective outcome	Monitoring the system, not just each agent in isolation (Ch. 8)

The failure modes worth actually expecting, plainly stated.

The one line to keep

Never leave an agent unsupervised on anything consequential — anything involving real money, a real customer, a real legal or safety obligation, or data you couldn't afford to lose or leak. That line doesn't move because a new model launched this quarter.

15 The honest bottom line

Where this leaves you, in one page.

An AI agent is an assistant you've given hands — able to act, not just advise. Used for narrow, supervised, reversible tasks, that's a real productivity gain you can start using this week. Handed unchecked power over anything that matters, it's a liability dressed as a shortcut, and the model doesn't behave any differently in the two setups — only your guardrails do.

Keep your finger near the stop button. Give it only the keys the task needs. Watch it work, and know exactly how you'd undo a mistake before you ever risk one. Score any vendor's pitch against Chapter 9's scorecard rather than their demo. Do that, and you get the upside — a genuinely useful few hours of legwork back every week — without betting the business on it.

Running agents costs real, metered compute — the honest infrastructure behind every tool call in this guide.

Source: Photo: a technician works on server hardware in an automated data processing room — U.S. Navy photo by MC3 Josue L. Escobosa, Wikimedia Commons, public domain

16 Image credits & attributions

Every photograph in this guide is real, licensed, and credited — nothing here is stock art pretending otherwise.

Sourced via the Wikimedia Commons API and used under the licences below. Full source URLs are recorded in the sources list on the following page.

WHERE IT APPEARS	SUBJECT	PHOTOGRAPHER / SOURCE	LICENCE
Cover	An emergency stop button, close up	Biso — Wikimedia Commons	CC BY-SA 4.0
The anatomy of an agent	A robotic arm's control panel, buttons and a screen, in an active factory	Shixart1985 — Wikimedia Commons	CC BY 2.0
The control chapter	A telescope operator at work in the VLT control room, Paranal Observatory	ESO — Wikimedia Commons	CC BY 4.0
A safe first agent to try	Rodney Brooks with Baxter, a collaborative robot designed to work safely alongside people	Steve Jurvetson — Wikimedia Commons	CC BY 2.0
The honest limits	An autonomous delivery vehicle pileup — robots stuck by their own proximity rule	Mbrickn — Wikimedia Commons	CC BY-SA 4.0
The honest bottom line	A technician working on server hardware in an automated data processing room	U.S. Navy photo by MC3 Josue L. Escobosa	Public domain

Photograph credits.

17 Sources, glossary & the ledger

Where every claim comes from — and what the plain words mean.

Sources

01 **Artificial Intelligence Risk Management Framework (AI RMF 1.0)** — US National Institute of Standards and Technology (as of July 2026)
https://www.nist.gov/itl/ai-risk-management-framework

02 **Guidelines for secure AI system development** — UK National Cyber Security Centre (as of July 2026)
https://www.ncsc.gov.uk/collection/guidelines-secure-ai-system-development

03 **Building effective agents** — Anthropic (as of July 2026)
https://www.anthropic.com/research/building-effective-agents

04 **Model Context Protocol — specification** — Anthropic / MCP open-standard contributors (as of July 2026)
https://modelcontextprotocol.io

05 **Everything is an agent — and the agents write their own tools** — vocabotics Lab Notebook (as of May 2025)
/lab/everything-is-an-agent

06 **File: Emergency Stop.jpg (cover photograph)** — Wikimedia Commons — Biso, CC BY-SA 4.0 (as of July 2026)
https://commons.wikimedia.org/wiki/File:Emergency_Stop.jpg

07 **File: Control panel of a robotic arm in a factory.jpg** — Wikimedia Commons — Shixart1985, CC BY 2.0 (as of July 2026)
https://commons.wikimedia.org/wiki/File:Control_panel_of_a_robotic_arm_in_a_factory.jpg

08 **File: Telescope operator at work in VLT control room (img-4458).jpg** — Wikimedia Commons — ESO, CC BY 4.0 (as of July 2026)
https://commons.wikimedia.org/wiki/File:Telescope_operator_at_work_in_VLT_control_room_(img-4458).jpg

09 **File: Rethink Robotics — Brooks and Baxter (8000143255).jpg** — Wikimedia Commons — Steve Jurvetson, CC BY 2.0 (as of July 2026)
https://commons.wikimedia.org/wiki/File:Rethink_Robotics_%E2%80%94_Brooks_and_Baxter_(8000143255).jpg

10 **File: Autonomous Delivery Vehicle Pileup.jpg** — Wikimedia Commons — Mbrickn, CC BY-SA 4.0 (as of July 2026)
https://commons.wikimedia.org/wiki/File:Autonomous_Delivery_Vehicle_Pileup.jpg

11 **File: US Navy 090902-N-9928E-009 ... automated data processing server room.jpg** — Wikimedia Commons — U.S. Navy photo by MC3 Josue L. Escobosa, public domain (as of July 2026)
https://commons.wikimedia.org/wiki/File:US_Navy_090902-N-9928E-009_Information_Systems_Technician_2nd_Class_Piotr_Zelaskowski,_from_San_Antonio,_Texas,_re-attaches_the_hard_drive_of_a_computer_in_the_automated_data_processing_server_room.jpg

Plain-English glossary

Agent	An AI system given a goal and the ability to act — using tools to search, read, draft, send or otherwise change something in the world — planning and adjusting its own steps rather than simply replying in words.
Tool / tool call	Anything an agent can call to affect the world outside its own reply — a web search, a file read, an email-send function, a database query. The tools it's given, and who decided that, is the single most important safety question.
MCP (Model Context Protocol)	An open standard, published by Anthropic, for how a model discovers and calls tools consistently across different products — worth knowing the name of when a vendor talks about their agent's 'tool use.'
Agentic loop	The repeating cycle an agent runs: act, observe what happened, decide whether to re-plan, act again — continuing until it believes the goal is met or it's stopped.
Guardrail	A deliberately built-in limit on what an agent can do or how far it can go without a human — least access, an approval gate, a hard cap on cost or steps, an audit log.

Least access / least privilege	Giving an agent only the specific permissions its current task needs, not broad standing access 'to save a step' — the single biggest lever on how much damage a mistake can do.
Human in the loop	A design where a person must review and approve an agent's output before a real, often irreversible, consequence happens — the difference between Chapter 5's two identical-model setups.
Hallucination	A fluent, confident output that is simply wrong — a made-up detail, an invented clause, a plausible-sounding plan that doesn't actually achieve the goal. Happens in agent plans exactly as it does in chatbot replies.
Compounding error	The arithmetic fact that reliability multiplies across a chain of steps — 90% per step becomes roughly 35% over ten steps — which is why a gate partway through a long run matters more than one at the very end.

vocabotics

THE HONESTY LEDGER

🟢 Shipped — measured, gate-clean.

🔬 Research — measured, partial; the gap is shown.

🔭 Vision — designed, honest; not yet built.

The bigger claim goes up a tier, never down. We show the eval beside the demo — the tiering is the credibility.

Read: Everything is an agent (Lab Notebook)

Once the anatomy and the guardrails make sense, this is the deeper, dated research view from our own lab — the same six-part agent shape, in production, with the honest limits of 'self-growth' kept on the page.

/lab/everything-is-an-agent

Nothing faked.

This guide was drafted with AI and verified by a named human. The worked run in Chapter 4 is illustrative — built to be exactly the shape of what happens with today's agents — and marked as such; the vendor scorecard is our own honest, dated judgement, not an independent benchmark; every photograph is real and credited on its own page.