

Build an AI agent without learning to code.

Ten free courses, in the order that actually works, plus what to build after each one. Four weekends, about twenty five hours, and one small agent at the end that does a real piece of your job.

BY DAVE BROWN

Most material on building AI agents is written by engineers, for engineers. It opens with a framework, a repository, and an architecture diagram, and gets around to explaining what the thing is supposed to accomplish somewhere near the end. If you are an operator, a marketer, a consultant, a founder, or a product manager, that order is backwards. You already know the work. What you are missing is the shape of the machine.

So this guide runs the other way. It starts with how to think about the problem, moves to tools you can use by dragging boxes around a screen, and only then goes anywhere near a technical detail. Every resource in it is free to study. Every stage ends with something you build.

The hard part was never the code. It is deciding what the agent is allowed to do.

That is the honest reason non-coders can do this now. The building got easy. The deciding did not, and the deciding is the part you are already good at. You know which requests are routine and which need a human. You know what a bad answer costs. You know which document is the one people actually trust. Nobody has to teach you that, and no platform can supply it.

What follows is a sequence, not a product ranking. You will not adopt ten platforms. You will use ten resources to learn one transferable skill: how an agent gets scoped, instructed, connected, grounded, tested, and improved.

10

free resources,
sequenced in six stages

6

parts every working
agent has

4

weekends, if you want a
schedule

\$0

to start. Free tiers cover
all of it.

WHAT'S IN HERE

- 01 What you're actually learning. The six parts, and the one distinction worth arguing about.
 - 02 The ten resources. Why each one is where it is, and what to build when you finish it.
 - 03 Four weekends. A schedule, and the question each one answers.
 - 04 The capstone. One small agent, six components, ten test cases.
 - 05 Where people get stuck. Symptom in, next step out.
-

PART ONE · WHAT YOU'RE ACTUALLY LEARNING

An agent is a job description for something that cannot ask you what you meant.

Six parts, one distinction, and an honest note about the two places this gets uncomfortable.

Start with the definition, because a lot of confusion downstream comes from skipping it. **An agent receives a goal, reads the situation, decides what to do next, and uses tools or information to finish the job.** An automation follows the path you drew for it. An agent has some discretion over the path it takes.

That distinction sounds academic until something goes wrong. When an automation breaks, it breaks in the same place every time and you go fix that step. When an agent goes wrong, it went somewhere you did not anticipate, which is both the entire value and the entire risk. Discretion is the feature you are buying. It is also the thing you have to bound.

A fair amount of what gets sold as an AI agent is an automation with a language model bolted onto one step. That is not an insult. Plenty of those are excellent, cheaper, and easier to trust. But you should be able to tell which one you are looking at, because it tells you what can go wrong and how much supervision it needs. By the end of this guide, you will be able to.

The six parts.

Every agent that works in the real world, on any platform, in any price bracket, has the same six components. Learn to see them and the platform differences shrink to vocabulary and button placement.

- **A task worth doing.** Narrow, recurring, and painful enough that someone notices when it gets done.
- **Instructions.** What it does, how it behaves, what it never does. This is writing, not engineering.
- **Knowledge.** The specific material it is allowed to read and answer from.
- **Tools.** The systems it can touch, and whether it can only look or also change things.
- **A human checkpoint.** The moment it stops and asks, and the rule for when that moment arrives.
- **A way to tell if it worked.** Test cases, a metric, or at minimum a person who reads the output for a week.

The ten resources ahead are sequenced because each one teaches a couple of these better than the rest. Anthropic and Botpress teach the task. Zapier and Make teach the tools. Voiceflow teaches the checkpoint. n8n teaches knowledge and error handling. Salesforce and Microsoft teach permissions and accountability. Flowise shows you the whole machine with the cover off.

THE SHORTCUT, IF YOU WANT ONE

When an agent misbehaves, one of the six is missing or vague. Nine times out of ten it is the checkpoint or the criteria for success, because those are the two nobody enjoys writing.

What you don't need.

Worth saying plainly, because the fear of these is what keeps most people from starting.

- **You do not need to write code.** Every building tool here is visual. You connect boxes and type instructions in English.
- **You do not need to pick the right platform.** You are learning a skill that transfers. The tool you finish on will probably not be the one you started on, and that is fine.
- **You do not need a budget.** All ten resources are free to study. The free product tiers are enough to build everything in this guide.
- **You do not need to understand how the model works.** You need to know what it is reliably good at, what it is quietly bad at, and where to put a human.

And the two places it gets uncomfortable.

Naming these in advance so you do not mistake them for evidence that you are the wrong person for this.

The first is around resource six. Words like API, JSON, credential, and error handling start appearing. None of them are conceptually hard. An API is a doorway one system leaves open for another. JSON is a way of writing down a record so a machine can read it without guessing. A credential is a key you store once. They feel hard because they arrive as unfamiliar vocabulary in the middle of a task you were previously enjoying. Slow down for one afternoon and they become unremarkable.

The second is the demo cliff. Your first agent will work beautifully on the example you built it for and fall over on the third real input. This is normal and it is the exact moment most people quit. It is also the moment learning starts, because everything after it is about handling the inputs you did not imagine. That is why every stage below ends with something you build rather than something you read.

In the order that keeps you from quitting.

Six stages. Each entry has why it sits where it does, what you should be able to do afterward, and the one thing to build before moving on. Everything here was verified free to study on August 17, 2026.

Two resources, one afternoon, no software.

RESOURCE 01

Anthropic: Building Effective Agents.

FORMAT

Editorial and technical primer

DIFFICULTY

Beginner concepts,
developer asides

TIME

One focused sitting

Read it here: [Building Effective Agents](#)

This goes first because it is the clearest mental model in the collection, and because its central argument is a subtraction. Most of what you want does not need an agent. Simple, composable patterns beat elaborate architectures, and autonomy should be added only where it earns its place. Starting here saves you from the most expensive beginner mistake, which is building something clever for a problem that wanted something boring.

It is an engineering article. There are code samples. You are allowed to skim them, and nothing in the argument depends on them. Read the definitions, the diagrams, the workflow patterns, and the design principles. Those are the parts you will still be using in five years.

What you should walk away with:

- The difference between a workflow and an agent, stated in your own words
- How a model, tools, instructions, and context combine into a system
- Why autonomy is a cost, not a feature, until it improves the task
- The common patterns: routing, parallelization, orchestration, evaluation
- Why more complexity does not produce a better system

Then do this. Pick one recurring process you already run, and separate the parts that are predictable from the parts that need judgment. A content process, for example, always collects a brief, researches a topic, drafts, reviews, and requests approval. The sequence never changes. The research and the drafting need judgment. Those two steps are where an agent might belong. The rest probably wants an automation, or a checklist, or nothing at all.

BUILD THIS BEFORE YOU MOVE ON

A one page map with three columns: fixed steps, judgment steps, and the tools or information each step needs. Keep it. It becomes the working brief for everything else in this guide, and you will revise it four times before the end.

RESOURCE 02

Botpress Academy: How to Build and Ship Your First AI Agent.

FORMAT	DIFFICULTY	TIME
Short beginner course	Beginner	About 35 minutes

Take it here: [How to Build and Ship Your First AI Agent](#)

Thirty five minutes, and it addresses the thing that sinks most agent projects. They rarely fail in the building. They fail because nobody could say, in one sentence, what the agent was for, who it served, and how anyone would know it was working. Then the demo impresses a room, nothing happens for six weeks, and the project quietly ends.

Botpress spends its time on exactly those decisions: choosing a task, choosing where the agent lives, choosing a model, defining oversight, handling compliance, picking a measure, and planning past the prototype. It is a course about scoping disguised as a course about building, which is the right way around.

What you should walk away with:

- A narrow first use case, chosen on purpose rather than by enthusiasm
- An honest answer to whether it needs an agent at all
- The right channel for the people who will use it
- A defined escalation path to a human
- A success measure simple enough to check weekly

Then do this. Turn the map from resource one into a project brief. Answer six questions and refuse to move on until each answer fits in a line: Who uses this? What single job does it do? What can it read? What can it change? When must it stop and ask a person? How will I know it worked?

BUILD THIS BEFORE YOU MOVE ON

A one page agent charter. Make the first version narrower than feels satisfying. Narrow scope is not a limitation you accept early on, it is the reason the thing ships at all. You can widen it later, once something is running and someone is using it.

Botpress Academy is free to study. The platform has a [pay as you go tier starting at \\$0](#) with limited usage and a small monthly AI credit.

STAGE TWO · BUILD A FIRST VISUAL AGENT

The weekend it stops being reading.

RESOURCE 03

Zapier Academy: AI Builder Path.

FORMAT	DIFFICULTY	TIME
Four course learning path	Complete beginners, explicitly	About two hours

Take it here: [AI Builder Path](#)

Zapier is the gentlest possible on ramp. The path starts with triggers, actions, field mapping, and app connections, which is the plumbing under every agent you will ever build, and only then introduces agents themselves. Zapier states outright that no coding and no prior Zapier experience is required. The final section covers giving an agent tools, instructions, and guardrails.

The reason to start here rather than somewhere more powerful is momentum. You need one thing that runs on its own, on real data, this weekend. That first moment where something happens in an app you did not touch is worth more to your learning than another two hours of theory.

What you should walk away with:

- How data moves between applications, and why field mapping is where things break

- A working trigger and action workflow
- Conditions and utility steps, so the path can fork
- A workflow described in plain language and built from that description
- An agent with a deliberately small set of tools and stated boundaries

Then do this. Build one agent that handles a low risk, multi step task. Good first projects: classifying inbound requests, qualifying leads, summarizing form submissions, preparing a meeting brief, routing customer questions, or drafting a reply that a person approves before it sends. Notice what those have in common. Every one of them either produces a draft or moves information. None of them can embarrass you on their own.

BUILD THIS BEFORE YOU MOVE ON

One working agent that completes a real task using at least two connected applications. Real means your data, your inbox, your spreadsheet. A sandbox demo will teach you the interface and none of the lessons.

Free with a Zapier account. Platform activity limits still apply.

RESOURCE 04

Make Academy: Automation to AI Agents, Foundation.

FORMAT

Six courses plus an assessment

DIFFICULTY

New to Make

TIME

Two to three and a half hours

Take it here: [Automation to AI Agents: Foundation](#)

Zapier teaches you to think in lines. Make teaches you to think in shapes. Information branches, loops back, passes through routers, and splits across applications, and you can see all of it on one canvas. That visual difference is not decoration. It is the first time the structure of your process becomes something you can look at and criticize.

The course also draws a distinction worth internalizing early: traditional automation, AI assisted automation, and agentic automation are three different things with three different risk profiles. Most real systems end up containing all three, and knowing which is which is how you decide where the supervision goes.

What you should walk away with:

- Foundational language model concepts, in business terms
- Visual workflows spanning several applications
- A working sense of when an agent beats a fixed automation, and when it does not
- An agent's objective, knowledge, and tools planned before it is built
- One agentic automation you built and ran yourself

Then do this. Rebuild your Zapier project in Make, and add one decision point. An inbound request agent might read the request, classify its urgency, retrieve relevant information, draft a response, escalate anything high risk, and log the result. Building the same thing twice feels wasteful. It is the fastest way to learn which parts of your first build were the idea and which parts were just the platform.

BUILD THIS BEFORE YOU MOVE ON

A visual diagram of that workflow, annotated. For every step, write whether it is fixed, AI assisted, or agentic, and one line on why. This is the exercise that turns the Anthropic reading into something you own.

The learning path is free. Make's free product tier supports limited experimentation.

STAGE THREE · DESIGN THE CONVERSATION

The first time a person is on the other side.

RESOURCE 05

Voiceflow Basics.

FORMAT	DIFFICULTY	TIME
Guided visual course	Beginner	One or two sessions

Take it here: [Voiceflow Basics](#)

Everything so far has happened backstage. Voiceflow turns the camera around and puts a person in front of the agent, which changes the design problem completely. It covers workflows, user input, intents, variables, conditions, knowledge, integrations, and conversational experience design.

Do this stage even if you never use Voiceflow again. Every agent that talks to a human needs a plan for the moment it does not understand, and most first drafts do not have one. They have a happy path and a cliff. What separates an agent people trust from one they route around is almost always the handling of confusion: does it guess, does it ask, or does it hand off?

What you should walk away with:

- A conversation with a deliberate structure rather than an accidental one
- Information captured from the user once and reused
- Branches and fallback behavior that degrade gracefully
- An agent grounded in a specific knowledge source
- Integrations to outside services, and a view on what happens when one fails

Then do this. Build a support or FAQ agent grounded in a small, trustworthy set of documents. A product information assistant, an onboarding guide, an event concierge, a policy assistant, a lead intake agent. Small and trustworthy matters more than comprehensive. Ten documents you have read beat four hundred you have not, because you can tell instantly when the answer is wrong.

BUILD THIS BEFORE YOU MOVE ON

A working conversational agent containing all five of these:

- A clear introduction that says what it can and cannot do
- Three supported user goals
- One clarifying question
- One fallback response for when it has no idea
- One human escalation path

Voiceflow offers a [free Starter tier](#) with limited features and credits, intended for experimentation rather than production.

Where it stops being a demo.

RESOURCE 06

n8n Quickstart.

FORMAT	DIFFICULTY	TIME
Hands on, with exercises and assessments	Beginner to early intermediate	Three to five hours

Take it here: [n8n Quickstart](#)

This is the stretch course, and it is the one that will make you feel out of your depth. That feeling is the point. n8n gives you considerably more flexibility than the platforms before it, and in exchange it stops hiding the machinery: APIs, JSON, credentials, data transformation, error handling. The course says coding experience is not required, and that is true, though familiarity with APIs and JSON helps. It ends with a customer service agent that can reach real data, hold memory, and call tools.

Two things make it worth the discomfort. The first is error handling, which nothing before this taught you and which is the difference between an agent you can leave running and one you have to watch. The second is that once you have seen how an agent reaches out to a system, retrieves a record, and answers from it rather than from memory, you will never again be confused about why agents make things up. They make things up when nothing told them where to look.

What you should walk away with:

- Triggers, nodes, and how data physically moves between steps
- A model connected to your business data
- Memory and tools, and the difference between them
- Information transformed and routed rather than passed along whole
- Errors and edge cases handled on purpose

Then do this. Build an internal information agent that looks something up before it answers. A customer, an order, a project, an inventory item. The behavior you are after is specific: the agent should retrieve the record first and then respond from what it found, rather than producing a confident sentence out of thin air.

BUILD THIS BEFORE YOU MOVE ON

An agent with all five: one external data source, one tool, memory, a defined error path, and a human approval step before anything consequential happens.

The course is free. Hands on work needs an n8n instance and a model provider API key. Cloud offers a trial; self hosting is free and more technical.

RESOURCE 07

Relevance AI Academy.

FORMAT	DIFFICULTY	TIME
No code tutorial and learning library	Beginner	One to two sessions

Take it here: [Build Your First No-Code AI Agent](#)

After the technical stretch of n8n, this is a deliberate return to easier ground with a new idea attached: more than one agent. Relevance AI is built around assembling specialized agents and tools into something like a small team, which makes it the natural bridge from one agent to several. It is explicitly no code, with templates and a natural language builder, so you can prototype a structure before you fully understand every part of it.

One caution, learned expensively by many people. Two agents are not twice as good as one agent. They are one agent, plus a handoff, and the handoff is where quality leaks out. If agent one summarizes its findings and agent two writes from the summary, agent two is working from a compressed version of reality and every subsequent step inherits the compression. Pass evidence, not conclusions.

What you should walk away with:

- An agent created from a description or a template
- Specialized tools attached to a specific role
- A business process translated into instructions someone else could follow
- A feel for how specialized agents collaborate, and where they misfire

- An early view on actions, cost, and whether any of this pays for itself

Then do this. Build a two role workflow. One agent gathers and summarizes, a second turns findings into a brief, a human reviews before anything is used. Lead qualification, market research, content repurposing, and support preparation all fit this shape.

BUILD THIS BEFORE YOU MOVE ON

The two role workflow, plus a written note for each agent stating what it is allowed to do and what it must hand off. If you cannot write that note, the roles are not actually separate and you have built one agent with extra steps.

Relevance AI offers a [free plan](#) with limited monthly actions and vendor credits. No credit card required as of this research.

STAGE FIVE · SEE HOW ENTERPRISES DO IT

Pick one. Whichever one your company already pays for.

RESOURCE 08

Salesforce: Agentblazer Champion 2026.

FORMAT

Structured Trailhead curriculum

DIFFICULTY

Intermediate, end user oriented

TIME

About eight and a half hours for the full trail

Take it here: [Become an Agentblazer Champion 2026](#)

Everything before this was you, building for you. This is the stage where the agent touches customer data and a business process, and a different set of questions arrives: whose data is this, which records may it read, what may it change, and who is accountable when it is wrong. Salesforce takes those seriously because its customers make it. The trail covers AI fundamentals, prompting, trust, Agentforce, Agent Builder, and a hands on service agent project.

It is product specific and it is the longest thing in this guide. Be strategic. If you do not work in Salesforce, prioritize Agentforce Basics and the first hands on agent build, and skip the rest without guilt. The transferable material is grounding, permissions, business rules, and trusted

actions, and you will have met all four within the first two hours.

What you should walk away with:

- The business case for an enterprise agent, stated the way a budget holder would state it
- How customer records ground an agent's answers
- Prompt templates, and why an organization standardizes them
- Why trust and data protection are design constraints rather than paperwork
- A basic service agent, assembled and connected to a real process

Then do this. Design an agent for a CRM shaped task: answering customer questions, assisting with bookings, qualifying an opportunity, summarizing an account, or preparing a service rep before a call. You do not need to build it. You need to be able to propose it.

BUILD THIS BEFORE YOU MOVE ON

A short enterprise proposal with six headings: business objective, required data, permitted actions, sensitive data restrictions, escalation rules, and success metrics. That document is the artifact that gets an agent approved. Everything else is a demo.

Trailhead is free and provides hands on environments for supported exercises.

RESOURCE 09

Microsoft Learn: Create Agents with Copilot Studio.

FORMAT	DIFFICULTY	TIME
Five module learning path	Beginner, app maker and business user	About two hours fifty minutes

Take it here: [Create Agents with Microsoft Copilot Studio](#)

The same enterprise questions from a different vendor, and considerably shorter. Copilot Studio emphasizes conversational authoring, knowledge sources, generative orchestration, variables, and Power Automate flows. If your organization runs on Microsoft 365, Teams, or the Power Platform, this is the more useful of the two enterprise stages by a wide margin, because the agent you design here can plausibly exist next month.

The reason both eight and nine are in this guide is not that you should do both. It is that whichever one matches your employer is the one that turns this from a hobby into something you can propose on a Tuesday. Do that one properly and skim the other for contrast.

What you should walk away with:

- A business oriented agent, planned before it is built
- An agent created through natural language description
- Structured conversation topics layered on top of that
- Answers grounded in approved internal knowledge rather than the open internet
- A connection to Power Automate, so the agent can actually do something

Then do this. Build an internal employee agent that answers from approved documentation and triggers one simple workflow. Finding an HR policy, preparing an onboarding checklist, submitting an internal request, looking up an approved procedure, routing a question to the right team. Internal agents are the best first enterprise project because your colleagues will tell you immediately and bluntly when the answer is wrong.

BUILD THIS BEFORE YOU MOVE ON

A prototype with one knowledge source, one action, and one escalation path. Three components, and it is a real system.

Microsoft Learn content is free. Full hands on work requires a suitable Microsoft environment; Copilot Studio offers a trial with account and publishing restrictions.

STAGE SIX · OPEN THE HOOD

The same machine, with the cover off.

RESOURCE 10

Flowise: documentation and tutorials.

FORMAT

Open source docs and tutorials

DIFFICULTY

Early intermediate

TIME

One focused session, then return as needed

Read it here: [Flowise Documentation](#)

Flowise comes last because it is the most open ended thing here and because it makes the most sense once you have already built five agents. It is a visual interface over assistants, chatflows, agent flows, retrieval, tools, evaluations, and human review. Everything the

managed platforms tastefully hid from you is on the screen. That is the value, and it is also the cost: installing, hosting, and connecting models introduces real technical friction that none of the previous nine had.

Start with the Assistant builder, which is the friendliest entry point, and let the rest wait. The concept worth taking away is retrieval, sometimes dressed up as RAG. Stripped of the acronym it means: look it up before you answer. That single behavior is the difference between an agent that sounds authoritative and one that is, and seeing it assembled by hand makes it permanent knowledge rather than a feature toggle you once flipped.

What you should walk away with:

- The components of an open agent system, named and recognized
- An assistant with instructions, tools, and knowledge, wired by you
- Retrieval, understood mechanically rather than as a buzzword
- A visual sense of more complex flows: branching, routing, multiple agents
- A clear view of the trade between open source flexibility and managed convenience
- Honest recognition of where you would want technical help

Then do this. Build a document assistant over a controlled knowledge base. When it works, add exactly one thing: a retrieval step, a human review checkpoint, a second specialized agent, a routing decision, or a simple evaluation. One at a time, so you can see what each addition costs.

BUILD THIS BEFORE YOU MOVE ON

An architecture diagram of that assistant, even if you never deploy it. If you can draw the system and explain each box to a colleague, you have the skill. The deployment is a separate problem and it is somebody's job, not necessarily yours.

Documentation and the open source software are free. Hosting and model usage may cost money and setup time.

A schedule, if you want one.

Each weekend answers one question. If you finish a weekend and cannot answer its question out loud, do that weekend again rather than moving on. The sequence compounds; the pace does not matter.

WEEKEND ONE

Mental models and the first working agent.

Complete: Anthropic, Botpress, Zapier.

Produce: a workflow map, an agent charter, and one simple agent that runs on real data.

THE QUESTION

What should this agent decide, and what should stay deterministic?

WEEKEND TWO

Orchestration and conversation.

Complete: Make, Voiceflow.

Produce: a branched visual workflow, a conversational agent with real fallback behavior, and a written list of the knowledge sources it is allowed to use.

THE QUESTION

How do information and people move through this system, including when something goes wrong?

WEEKEND THREE

Data, tools, and collaboration.

Complete: n8n, Relevance AI.

Produce: an agent connected to real or sample data, a tool calling workflow, a two role agent system, and a human approval checkpoint.

THE QUESTION

What can this agent see, and what is it allowed to change?

WEEKEND FOUR

Enterprise systems and open architecture.

Complete: selected modules from Salesforce, Microsoft, and Flowise. Selected is the operative word. This weekend is the only one where skipping is the correct strategy.

Produce: an enterprise use case proposal, a permissions and escalation plan, a lightweight evaluation checklist, and an architecture diagram.

THE QUESTION

What would make this safe, measurable, and maintainable by someone other than me?

PART FOUR · THE CAPSTONE

One small agent, around a process you actually run.

Not a portfolio piece. Something that removes a genuine annoyance from your week, which is the only kind of agent anyone keeps using.

Good candidates, all of them narrow on purpose: a research and briefing agent, a content intake and drafting agent, a lead qualification agent, a support preparation agent, an employee policy assistant, a meeting preparation agent.

Whichever you choose, it needs six components, which are the same six from part one, now with your name on them.

1. **A defined user and a single task.** One person, one job. Not a platform.
2. **Written instructions.** Including what it must never do, which is the half people skip.
3. **A limited knowledge source.** Limited is the important word. You have to be able to check its answers.
4. **At least one tool or connected application.** Otherwise it is a chat window with a personality.
5. **A human approval or escalation point.** With a stated rule for when it triggers.
6. **Ten test cases.** Written before you finish building, not after.

About those ten test cases.

This is the component that separates people who built an agent from people who can build agents, and it takes about twenty minutes. Write the cases before the agent is finished, so the standard cannot quietly bend to fit whatever you happened to produce.

- **Five ordinary cases.** The requests you built it for. These should pass on the first run, and if they do not, nothing else matters yet.
- **Three awkward cases.** Vague phrasing, missing information, two questions at once, a request that is nearly but not quite in scope. This is where you find out whether it guesses or asks.
- **Two cases it must refuse.** Something outside its remit, and something that requires a human decision. An agent that never declines anything has not been given boundaries, it has been given permission.

Run all ten. Write down what happened. Then fix the instructions rather than the test, which is a temptation more real than it sounds at eleven on a Sunday night.

THE BAR TO CLEAR

Not perfection. A useful capstone gets the five ordinary cases right, asks a sensible question on the awkward ones, and refuses the two it should refuse. That is a working system, and it is more than most agent pilots inside large companies manage.

PART FIVE · WHERE PEOPLE GET STUCK

Symptom in, next step out.

Choose by what is actually blocking you, not by which stage sounds most advanced.

WHAT'S HAPPENING

GO HERE

I don't know what to build

Botpress, then write the charter. The six questions do the work.

I've read a lot and built nothing

Zapier, tonight, one hour. Momentum beats the next article.

WHAT'S HAPPENING

GO HERE

It worked in the demo and broke on real input

Voiceflow. You are missing clarification and fallback, not intelligence.

It answers confidently and it's wrong

n8n or Flowise. It has no grounded source, so it is producing plausible text.

It does one thing and I need five

Relevance AI, two roles. Watch the handoff, that's where quality goes.

It's good and I still don't trust it

The ten test cases, and one human checkpoint before consequential actions.

My company will never approve this

Salesforce or Microsoft, whichever they already pay for. Write the proposal, not the demo.

I want to see how it works underneath

Flowise, Assistant builder first. Add one component at a time.

I keep switching platforms

Stop. The platform was never the skill. Finish something on the one you're on.

THE WHOLE THING, COMPRESSED

You are not learning ten platforms. You are learning to write a job description.

Scope it narrowly. Write the instructions, including what it must never do. Give it a small body of knowledge you can check. Give it the fewest tools that get the job done. Decide in advance when it stops and asks a person. Then test it against ten cases you wrote before you were emotionally invested in the answer.

Every platform in this guide will look different in two years. Some will not exist. That sequence will be exactly the same, because it is not a fact about software. It is a fact about delegating work to something that cannot read your mind, will not ask a clarifying question unless you tell

it to, and will do precisely what you said rather than what you meant. You have delegated work to people. This is that, with the ambiguity removed by you, in advance, in writing.

ONE MORE THING

If you want a second read on your charter.

The one page from resource two, before you spend a weekend building on top of it.

Most agent projects I see were not badly built. They were built confidently on a scope that was too wide, with no stated escalation rule and no definition of success, and every hour of building after that made the problem more expensive to fix. The charter is ninety seconds to read and it is where almost all of the avoidable trouble lives.

So I am doing a small number of informal reviews. Send me your one page charter and a sentence on what you are planning to build it with. I will tell you where the scope is too wide, which of the six components is missing, and which stage in this guide I would do next. No pressure either way; everything above is yours to use regardless.

GET IN TOUCH

Email hi@davecto.com with the subject line **"Agent Charter"** and your one page attached.

More guides like this one, for people trying to use AI without embarrassing themselves.

Weekly, plain-language breakdowns on Instagram.

[@davecto](https://www.instagram.com/davecto)

A note on sourcing: course formats, durations, difficulty levels, and free tiers were verified on August 17, 2026, and every one of them is the kind of thing a vendor changes without announcing it. Check the pricing page before you plan around a free tier. The sequencing, the six components, and the ten test cases are working practice rather than published research. The claim I would most like you to check yourself is the central one: that scoping is harder than building. You can test that in a single weekend with resource three and one real task, and I would rather you did than took my word for it.