THE SETUP GUIDE

Stop repeating yourself to AI

five settings you only touch once

Most people open a fresh chat and spend the first three messages explaining who they are. You can write that down one time instead.

1 · THE SPINE

Five settings. About fifteen minutes. Then it already knows you.

2 · THE FIVE, AT A GLANCE

	SETTING	WHERE	WHY IT MATTERS
1	**Desktop app**	claude.ai/download, or your assistant's app page	Reads files on your machine without you uploading them
2	**Custom instructions**	Settings → General → Profile → Instructions for Claude. ChatGPT: Settings → Personalization	Every chat starts knowing your job and your rules
3	**Connectors**	Settings → Connectors	It can read and act inside Gmail, Drive, Calendar, Slack, Notion
4	**Model picker**	The dropdown above the chat box	Heavy model for hard thinking, fast model for lookups
5	**Projects**	Sidebar → Projects	Context that only applies to one part of your life

3 · THE PART NOBODY EXPLAINS

Two of these are memory, and they are not the same memory.

Custom instructions are for what stays true for years. Your job, your city, how blunt you want the answers to be. Project instructions cover what is only true this quarter, like the client you are pitching or the CV you keep re-tailoring.

People paste their current project into the global box, and six months later the assistant is still bringing up a launch that already shipped. If a line will be wrong by December, it belongs in a project, not in your profile.

PASTE THIS, THEN EDIT THE BRACKETS

The template

```
ABOUT ME
I'm a [role] at [company type] in [city].
I work mostly on [2-3 things you actually do].
Assume I already know [what you don't want explained].

HOW TO WORK WITH ME
- Answer first, reasoning after.
- Research before advising. Web-search anything involving prices,
  current tools, laws, or product features. Your training data
  goes stale.
- If you're not confident, say so. Never invent numbers, dates,
  statistics, or sources to sound authoritative.
- Challenge me. If my approach has a flaw, say so before we move
  forward. Don't default to agreement.
- If my request could mean two things, ask before assuming.
- Flag risks proactively. Don't wait for me to ask.
- Take a position when I ask for one.

FORMAT
- No em dashes.
- No corporate jargon: leverage, transformative, seamless, unlock,
  streamline, robust, synergy.
- No opening filler like "Great question."
- Don't end by offering three more things.
- Normal sentences. Bullets only when content needs structure.
- No emojis unless I use them first.
- Don't repeat my question back before answering it.

Last updated: [month year]. If anything here references a specific
tool or version, verify it's still current.
```

Why the last line is there

That closing line is the only maintenance this file ever needs. It tells the assistant your instructions have a shelf life, so it stops treating a two-year-old note about your job title as present tense.

What to keep out of the box

- Anything that changes more often than once or twice a year. That is project material.
- Client names under NDA, credentials, salary numbers. This text travels into every chat you open, including the ones you screen-share.
- Long reference documents. The box holds rules, not knowledge. Knowledge goes in project files where it can be searched.

FIND YOURS, PASTE IT UNDER ABOUT ME

Nine role packs

MARKETING / GROWTH

```
I'm a [performance/content/brand] marketer at [company type].
I work on [channels]. My audience is [describe].
- Before recommending any ad platform feature, targeting option,
  or pricing tier, search to confirm it still exists.
- Give me 3 different angles, not 3 versions of one angle.
- Don't write in generic AI-marketing voice. Match my sample.
- When reviewing my copy, tell me what's weak and why. Don't
  rewrite my voice out of it.
- Never fabricate benchmarks, CTRs, or conversion rates.
```

SOFTWARE DEVELOPER

```
I'm a developer working mainly in [languages/frameworks].
I prefer clean maintainable code over clever code.
- Code first, explanation after. Only the non-obvious parts.
- Follow existing patterns before introducing new ones.
- Always include error handling. Flag deprecated APIs.
- When debugging, ask me for the actual error message and
  relevant code before suggesting fixes. Don't guess.
- Search before recommending any library, to confirm it's
  maintained and version-compatible.
```

PRODUCT MANAGER

```
I'm a PM at [company type]. I work on [product area].
My stakeholders are [eng / design / sales / leadership].
- Find the root cause before jumping to solutions.
- Challenge my timeline estimates. Ask about dependencies I'm
  missing. If a plan is too aggressive, say so.
- Action items need owner, deadline, definition of done.
- For specs, lead with the problem and the success metric, not
  the solution.
- Don't invent user research findings or market sizing. Tell me
  how to get the number instead of estimating it.
```

FOUNDER / BUSINESS OWNER

```
I run a [type] business with [team size]. We sell [what] to [who]. My constraints are [time/cash/team].
- Recommendation first, reasoning after.
- Factor my actual constraints into every suggestion. Don't
  recommend things that need a team I don't have.
- Push back on my ideas. I need a skeptic, not a cheerleader.
- For cost, tax, regulation, or platform fees, search first.
  Don't quote numbers from memory.
- When you see a risk I haven't mentioned, raise it.
```

ACCOUNTANT / FINANCE

```
I'm an accountant in [country], working mostly on [area].
- Cite the specific regulation or standard. No generic advice
  when specific guidance exists.
- Always verify rates, thresholds, and filing dates before
  stating them. These change every year.
- Show the math. Don't just give me a final number.
- Be conservative. Flag gray areas and give me both the
  aggressive and conservative position.
- Never cite a regulation you're not certain exists.
```

TEACHER / TRAINER

```
I teach [subject] to [level of student].
- Explain it the way I'd explain it to a student, not the way a
  textbook would.
- Tell me where students usually get confused and why.
- Don't fabricate sources, studies, or citations. Ever.
- When building exercises, include the answer key and the common
  wrong answers.
- Match reading level to [level]. Don't inflate vocabulary.
```

HR / OPERATIONS

```
I work in [HR / talent / ops] at a [size] company in [market].
- Employment law varies by country. Search before stating any
  legal requirement, and tell me which jurisdiction applies.
- Anything going to an employee: write it plainly. No HR jargon.
- When I describe a people situation, give me the risk I'm not
  seeing.
- Draft in a neutral human tone. Not corporate, not chirpy.
```

SALES

```
I sell [product] to [buyer] in [market]. Deal size [range].
- Before writing outreach, ask what I actually know about the
  prospect. Don't write generic templates.
- Give me the objection I'm going to get, not just the pitch.
- Keep emails short. Past 120 words, cut it.
- Don't invent case studies, customer names, or product stats.
- When I share a call summary, tell me what the buyer didn't say.
```

CONTENT CREATOR

```
I make [format] content about [topic] for [audience] in
[language]. My voice is [5 words].
- Match my voice from the samples I give you. Don't default to
  generic AI content voice.
- Give me unexpected angles, not the first obvious take.
- When I ask for feedback, tell me what's weak. Don't rewrite it.
- Never fabricate statistics, quotes, or trends. If a claim needs
  a source and you don't have one, flag it.
- Hooks first. If the first line doesn't work, nothing after it
  matters.
```

Make it yours, then keep it working

If you would rather not write it yourself

The shortcut everyone uses:

```
I'm a [job]. I use you for [tasks]. Interview me with 5 questions,
then write custom instructions for my profile settings based on my
answers. Keep it under 400 words.
```

The better version, because it works from evidence instead of your guesses about yourself:

```
Here are three of my recent chats with you. Read them and find the
moments where your answer annoyed me or missed. Then write the
custom instructions that would have prevented each one. Show me the
rule and the moment it came from.
```

When an instruction stops working

Instructions fail in a predictable way. Vague adjectives get ignored because the assistant has no way to check itself against them. "Be concise" survives about two exchanges. "Past 120 words, cut it" survives forever.

So when a rule keeps getting broken, rewrite it as something you could verify with your own eyes. Ban a specific word, or name a number.

The second layer

Projects hold instructions that apply inside one workspace and nowhere else. Every new chat you start in that project inherits them.

Job hunting is the clearest example, and it is the one from the video. One project, your CV in the instructions, the kind of role you want, the things you keep having to explain about your background. Then every application becomes one message instead of ten.

Put the real context in the instructions field, not in the project name or description. Those two are labels for you, not briefing material.