EVERYTHING YOU NEED

The Complete Claude Handbook

for SMEs — From your first conversation to an AI that runs your business

Everything you need.

Contents

Why this handbook

You know you need to do something with AI. You hear it at every dinner, you watch competitors lean into it, and somewhere a quiet worry nags that you are falling behind. But every explanation you find is technical, abstract, or breathless hype, and by the end you still do not know what to actually do on Monday morning. This handbook was written to take that feeling away.

In plain language it walks you from your very first conversation with Claude to putting it safely to work on real customer-facing tasks, and even to an AI that knows how your business runs. Written for small and midsize businesses, with examples from the real world instead of from a conference stage. No hype, no jargon, just an honest and complete picture of what works, what it is worth, and where the limits are.

Who this book is for: two tracks

This handbook serves two readers at once, and you are probably both of them.

- The owner, who decides, sets things up, and keeps them safe. For you it is about the big calls: where to start, what it returns, how to stay secure, and how to bring your team along.
- The user, who works with it every day. For you it is about the concrete: how to write a good instruction, how to draft a document or a quote faster, how to use it in your role.

Both tracks run through the whole book. An owner skims some chapters at a high level; a hands-on user dives into the practice. Take what fits you.

The ladder: from first conversation to an AI that knows your business

Putting AI to work is not a switch you flip, it is a ladder you climb. At the bottom is the simple conversation: you ask, you get an answer. One rung up, you have Claude create documents and analyze files. Above that come Projects and knowledge bases that let it remember your context, Skills that teach it your standard ways of working, and scheduled tasks that make work run on its own. At the top is building: with Claude Code and with connectors and agents that work inside your own systems, under your supervision.

You do not have to climb all the way, and most people never do. But once you know the ladder exists, you understand where this is heading: from "I ask a question" to "I have an AI that knows my work." This book follows that ladder, and closes with what it means for each role and how to plan your first thirty and ninety days.

What makes this book different

There is no shortage of AI guides. This one earns its place on a few points that matter for a smaller business. It is built for SMEs, with examples from real industries. It takes data safety seriously, with a full chapter on handling customer data rather than a single throwaway line. It is honest about the limits, about hallucination, and about the mistakes everyone makes. And it is built to use: nearly every chapter ends with ready-to-paste, fully worked instructions you can copy and adapt, plus a "do this now" to get moving immediately.

About the author

My name is Tico van Gerner. I work as a Customer Experience and AI Automation Specialist, helping small and midsize businesses put AI to practical, safe use in customer contact, marketing, sales, and service. This book is not a pile of tips scraped off the internet. It is the distillation of years of using AI professionally, combined with deep research done specifically for this handbook. I write plainly, because I believe AI should be accessible to a small business, not dressed up as something complicated. What you read here is what I have seen work in practice, and just as important, what I have seen fail.

How to read this book

You do not have to read it cover to cover. Start at Part 1 if you are new, or jump to the chapter for your role if you want a focused start. Read with a notepad nearby, or better, with Claude open beside you so you try the examples right away. And remember the thread you will see in every chapter: the speed comes from the machine, the judgment stays with you. Keep your hand on the controls and you gain a tireless colleague.

Ready? Then start on the bottom rung. We build it up together.

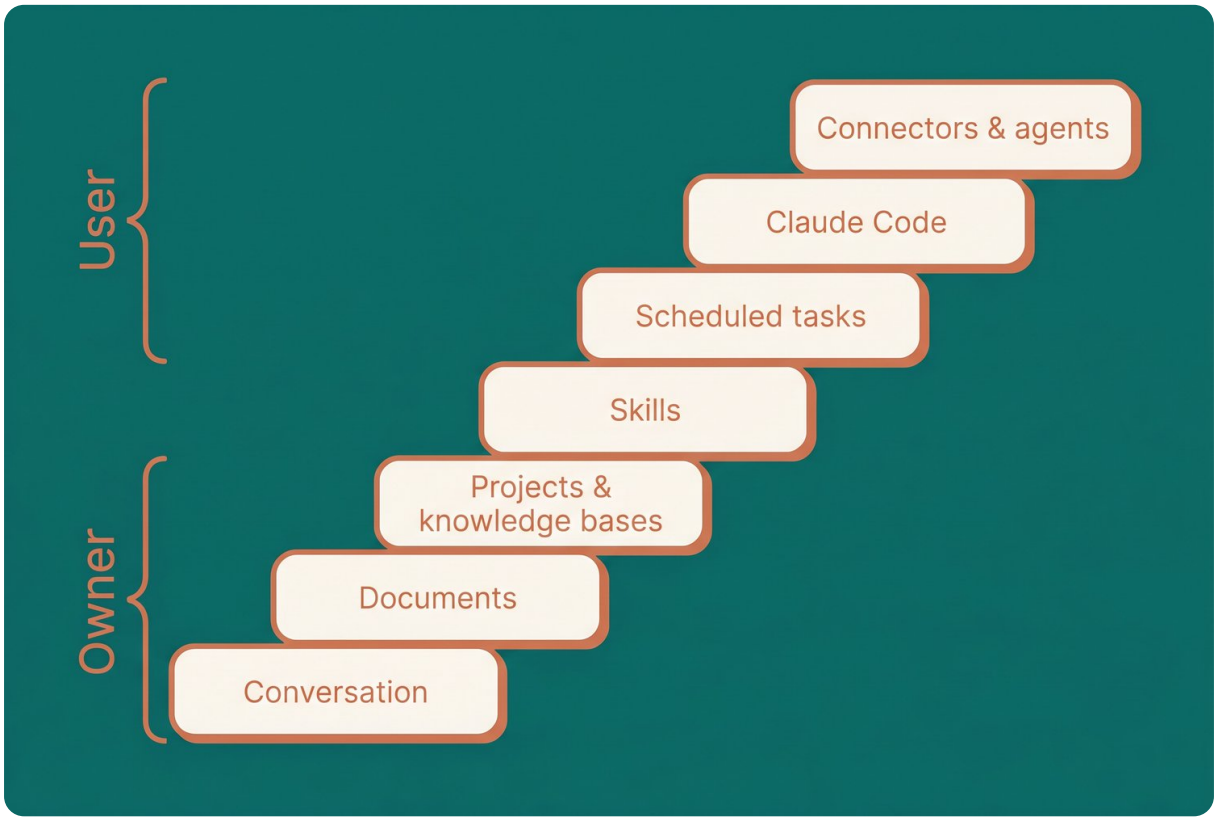


Figure - the ladder as the central visual, bottom to top: conversation -> documents -> projects and

What Claude is (and what it is not)

The first time someone opens Claude, they type three words, like in Google. "AC prices 2026." The answer is underwhelming, and the conclusion comes fast: AI is not for me. The mistake is not in Claude. It is in the picture you have of Claude.

This chapter gives you the right picture. Not because it is nice to know, but because it decides whether you get value out of it. Treat Claude like a search engine and you get disappointing search results. Treat it like a sharp colleague and you get a sharp colleague. That difference is bigger than any button or trick, and it is exactly where most people give up before it gets interesting.

A colleague, not a search bar

Claude is an AI assistant from a company called Anthropic. You give it an instruction in plain language, and it answers with considered text. The difference with Google is fundamental: a search engine looks up existing pages and hands you links, Claude thinks along and delivers you an answer. It writes, summarizes, structures, does the math, translates, compares, and reasons through a problem, all in one running conversation where you can steer.

Anthropic trained Claude to be three things at once: helpful, honest, and harmless. That is not marketing, you notice it in practice. It asks follow-up questions when your instruction is unclear. It tells you when something is off or not possible. And, when you set it up well, it does not blindly go along with an assumption that makes no sense. That is what makes it useful as a sparring partner, not just a text machine.

Four mental models, three of them wrong

To use Claude well you have to unlearn three stubborn misconceptions.

It is not a search engine. By default Claude does not pull live information off the internet. It works from what it learned during training, with a knowledge cutoff in the past. Ask it for today's price, your supplier's schedule, or this morning's news, and it can be wrong, unless you turn on web search (chapter 9). A lot of disappointment starts here: people expect Google and get something else.

It is not a database. A common misconception is that Claude is a giant storage box of ready-made answers it looks up. That is not how it works. It generates answers based on patterns it learned (chapter 2 goes deeper into this). That explains both its strength and its weakness: it can talk fluently about almost anything, but it can also say something fluent that is wrong.

It is not an oracle. Claude can state something false with full confidence, and on top of that it has a natural tendency to go along with you. Ask it "is it true that...", and there is a good chance it agrees, even when it should know better. Researchers call that sycophancy. The remedy is simple and it is in your hands: ask for pushback. "Play devil's advocate" or "what is wrong with my assumption" surfaces more honest answers than "do you think this is good?".

What it actually is: a colleague who thinks along. One who has read almost everything once, delivers a first draft in seconds, and patiently keeps refining. But also one who does not know your business until you tell it, who is sometimes too confident, and who you have to contradict now and then. Treat it that way, and it behaves accordingly.

Not a search bar. A colleague.

Where it shines

Anything to do with language and thinking. Concretely:

- Writing and rewriting: a messy thought into a clear email, a quote, a job posting.
- Summarizing: a long report, a contract, or an hour of notes boiled down to the core.
- Structuring: loose ideas organized into a plan, an approach, a checklist.
- Translating and shifting tone: the same text formal, casual, or in Spanish or German.
- Brainstorming: ten names, ten angles, ten objections a customer might raise.
- Explaining: a complicated topic at exactly the level you ask for.
- Searching and organizing your own text: hand it ten customer emails and ask which ones are about the same thing.

The thread: Claude is at its best when it works with language you supply or request, and when a human checks the result.

Where it is weak

Facts it cannot know. The exact sales-tax treatment of an edge case, your supplier's hours, the figure from a report you did not upload. There it can produce plausible-sounding nonsense. Current events fall under this too, unless web search is on. And it is sensitive to the quality of what you give it: garbage in, garbage out. The fix is always the same: give it the source, or let it do the reasoning and verify the fact yourself.

The same difference three times, from three angles:

- An HVAC installer in the Netherlands types "AC prices." He gets generalities. The same installer pastes his price list and says: "Estimate a ballpark price for two Daikin units on the second floor, 40 feet of line set." Now he gets a grounded estimate.
- A marketer asks "write something about our service." He gets slick, hollow text. The same marketer gives the audience, an earlier post that did well, and the goal. Now he gets something that sounds like his brand.
- A bookkeeper asks "what is the Section 179 deduction limit." Risky, because the answer may be out of date. Smarter: let Claude structure the question and draft a plain-language explanation, and verify the exact figure with the IRS or your CPA. The thinking is Claude's, the fact you verify.

The pattern: Claude is only as good as what you give it. That is chapter 3, and it is the single most important lesson in this book.

"That's for big companies"

One last misconception, and maybe the most expensive. Many owners think AI is for corporations with a data department. The opposite is true. It is precisely in a small business, where everyone wears too many hats, that Claude takes away the work nobody gets to: the quote that keeps slipping, the email that takes too long, the report no one reads. You do not need an IT department. You need an account and half an hour to capture your way of working once.

When not to use Claude

Being honest about the limits is part of it, and it makes your use better.

- For a final legal, medical, or financial judgment without a human checking it. Use it to prepare, not to decide.
- For facts that must be exact and that you cannot verify. No source, no certainty.
- With sensitive customer data in a free personal account. That is chapter 7, and it is the first mistake most businesses make.
- As the only voice in a decision. It is a thinking partner, not a decision-maker; you keep the judgment.

Pro tips from practice

- Treat it like a smart colleague, not like Google. You give a colleague context, not keywords.
- Give it a role. "Think like a skeptical customer" or "you are an experienced buyer" steers the answer more than ten adjectives.
- Ask for pushback. Because of its tendency to agree, you get better answers with "what is wrong with my plan?" than with "is this good?".
- Turn on web search when you need current facts. Otherwise do not let it guess at today's news.
- Check every number, name, and source as if an intern supplied them. Often it is right. Sometimes it is not.
- Use it in reverse too: have it explain something you already have. Paste a contract and ask what it says.
- One question per conversation works better than five at once.

Common mistakes and how to avoid them

- Treating it like a search engine. Fix: give context and an instruction, not keywords.
- Expecting it to know your business. Fix: tell it who you are and paste an example.
- Blindly agreeing with what it says, or letting it agree with you. Fix: actively ask for pushback.
- Copying numbers on faith. Fix: ask for the reasoning and verify the figure.
- Quitting after the first mediocre answer. Fix: steer it. The second attempt is almost always better.

Ready to use: set Claude up as your colleague

Filled in here as an example. Paste it at the start of a conversation or save it as a project instruction (chapter 10), and replace the details with yours:

```
You are my thinking-partner colleague for my HVAC company in the Netherlands
(five technicians, mostly residential customers).
Context about me: I write plainly and concretely, no sales fluff.
My customers want clear explanations and a quote without jargon.

How to work:
- Think briefly before you write.
- Ask follow-up questions when information is missing; do not make things up.
- Be honest when something is off or when you cannot know it.
- Do not automatically agree with me. Push back when my assumption is shaky.
- Do not give facts or numbers you are not sure of; say when you need web search.
```

Claude next to ChatGPT, honestly

You do not have to choose, but it helps to know where each one excels. The honest state of things in 2026: the top models are close, on most tests it is a few percent either way. The difference is in specialization.

- Claude excels at natural writing, long documents (it holds far more text at once), careful reasoning, and collaborative refining. It also more often says honestly "I am not sure about this" instead of confidently saying something wrong, which makes it popular in legal, financial, and healthcare settings.
- ChatGPT is broader: image generation, voice, and a large palette of ready-made integrations. For someone who wants one all-rounder for odd jobs, that is convenient.

A common piece of advice from people who use both daily: use both, each for its strength. For writing, analysis, and project work, Claude; for images and broad odd jobs, something else. One nuance that keeps coming back: Claude rewards people who know what they want. Give it direction and context and you get a partner; throw in three words and you get an obedient text machine.

Do this now

Take a question you would type into Google today. Instead, give Claude three sentences of context, a clear instruction, and a role. Add one line: "also name what could be wrong with my approach." Compare the answer with what Google would give you. That difference is what this book is about.

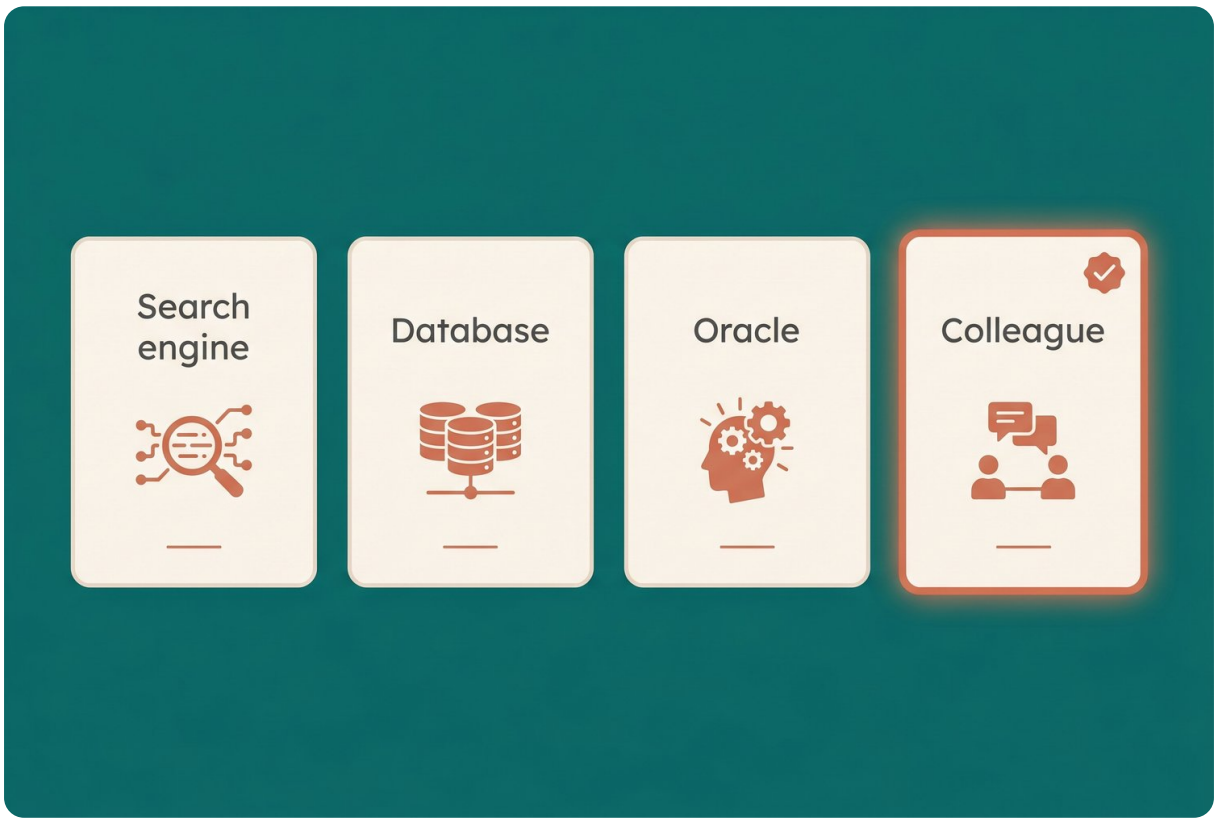


Figure - four cards

In the next chapter we look under the hood, in plain language: how Claude arrives at an answer, why it sometimes invents something with full confidence, and why it tends to tell you you are right.

How it works, in plain language

Ask Claude about a book that does not exist, and it will write you a convincing summary, with main characters and a plot. No bad intent. That is simply how the thing works. And precisely when you understand why, you know exactly when you can trust it and when you have to check.

You do not have to become a techie. But these few ideas under the hood change how you use Claude, and they explain almost every frustration beginners have.

It predicts language, piece by piece

Claude was trained on an enormous amount of text. From that it learned patterns: which word, which sentence, which idea logically follows from what is already there. Ask it something, and it predicts the most fitting continuation step by step. Think of it as autocomplete, but a hundred times more powerful and with an understanding of coherence.

Two misconceptions are important to unlearn here. It is not a database that looks up ready-made answers; it generates every answer fresh. And it does not follow fixed rules like a decision tree; it works with probability. It keeps choosing the most likely continuation, given your question and everything it ever saw. That explains why two nearly identical questions sometimes give slightly different answers, and why small changes in how you ask make a difference. It is not a broken device being inconsistent; it is a probability machine, and that is how it is supposed to behave.

It thinks beyond a single word

A stubborn misconception is that it just muddles along, word by word, with no overview. Research from Anthropic shows something else. Claude thinks ahead: it has an idea of where a sentence or paragraph should go and writes toward it. It also works in concepts, not in loose words of one language, and only pours that understanding into Dutch or English at the end.

That explains why the answers feel coherent and considered, not like loose words glued together. There is structure behind it. It also means it genuinely reasons about your question, and does not just parrot text, which is exactly why it can be such a good sparring partner when you feed it well.

Why it sometimes makes things up

Because it generates probable text instead of looking up facts, it can produce something that is right in form but wrong in fact. That is called hallucination. A few real examples:

- You ask for a source, and you get a tidy title with author and year that does not exist.
- You ask for a specific section of a law, and you get a plausible number that is wrong.
- You ask for a figure ("what percentage..."), and you get a round, convincing number with no source.
- You ask for a product's specs, and it fills missing details with plausible inventions.

The thread: hallucinations sound just as confident as the correct answers. You cannot hear it in the tone. You recognize it by the type of question. Facts, figures, names, quotes, and sources are the risk zone; explanation, structure, and ideas are far less so.

Sounds certain is not the same as is correct.

Why it often tells you you are right

There is a second pitfall that is less known but just as important. Claude was trained to be helpful, and that sometimes tips into being a pushover: it tends to agree with you. Ask "is my approach right?", and there is a good chance it confirms. Ask "this is a good idea, right?", and you get applause. For a beginner that is dangerous, because it feels like validation while it is mostly politeness.

The remedy is in your hands: actively ask for pushback. "What is wrong with this plan?", "Play devil's advocate", "Name three reasons this fails." Same question, different framing, and you get a more honest and more useful answer. Treat agreement with healthy suspicion; treat criticism you invited yourself as gold.

Your fact-check routine

You do not have to distrust everything. You only have to know when to check. A routine that works:

- Ask for the source. "Where does this come from?" If the source stays vague, it is suspect.
- Give the data yourself. Upload the report or the price list instead of asking whether Claude knows it by heart.
- Turn on web search for current facts, so the answer gets a verifiable source.
- Ask about uncertainty and counterarguments. "What are you unsure about?" and "what argues against this?" surface mistakes.
- Scale your checking to the stakes. An internal brainstorm does not need it, a quote to a customer or a figure in a report does.

A word about memory and context

Claude remembers everything said within one conversation; that is called the context window. Across conversations it remembers nothing by default, unless you use memory or a Project (chapters 9 and 10). Two practical consequences. One: start a new topic in a new conversation, otherwise old context keeps interfering. Two: in very long conversations the attention thins out, because there is so much in the window that the important instructions get buried. So keep a conversation focused, and repeat important instructions when a conversation gets very long.

Pro tips from practice

- Give it the facts, do not ask it to recall them. It is strong at processing, weak at remembering.
- Ask for the reasoning, not just the outcome. Then you see where a mistake creeps in.
- Invite pushback instead of confirmation. You want to know what is wrong, not hear that it is fine.
- High stakes always means checking; low stakes can go on instinct.
- Keep conversations short and focused; start a new conversation per topic.

Common mistakes and how to avoid them

- Trusting the confident tone. Fix: judge by the type of question, not by how assertive it sounds.
- Letting it agree with your own assumption. Fix: explicitly ask for counterarguments.
- Asking for facts it cannot know. Fix: give the source or turn on web search.
- Endless conversations where everything runs together. Fix: new topic, new conversation.

Ready to use: a source-aware, critical way of working

Paste this for fact-sensitive or important work, or save it as a project instruction (chapter 10):

```
For facts, figures, names, and quotes:
- Say when you are not sure; do not invent a source.
- State where your information comes from, or say you cannot verify it.
- If I ask for current or exact data, flag that web search or a
  source is needed.

For my plans and assumptions:
- Do not automatically go along with me. Name what is wrong or missing.
- Close with: what are you least sure of, and what argues against it?
```

Turning off the people-pleasing: concrete tricks

In chapter 1 you saw that Claude tends to tell you you are right. Power users share a few tricks that genuinely work, and they cost nothing:

- Set it in your settings. Add a standing line to your profile instructions: "Be honest, not nice; push back when I am off." Then it applies in every conversation.
- Repeat the rule in a few places in a long instruction. Claude weighs an instruction it meets several times more heavily; it infers that you really mean it.
- Frame it as someone else's work. Do not ask "is my plan good?" but "critically assess this colleague's plan." Claude then feels less of an urge to please you.
- Ask truth-first: "do not soften hard truths to spare my feelings."

A ready-to-use block, filled in here; paste it at the top or put it in your settings:

```
Be honest, not nice. Your job is not to agree with me, but to make
me sharper.
- Start with what is wrong or missing, then what is good.
- Do not soften hard truths; I want the real assessment.
- Name at least two risks or counterarguments to my idea.
- In doubt? Say so, and do not invent facts.
```

This one block turns Claude from a yes-man into a sparring partner.

Do this now

Ask Claude a question you already know the answer to. Then ask two things: "What are you uncertain about, and how can I verify this?" and "What would a critic argue against your answer?" You will see at once how differently it behaves when you ask for honesty and pushback, and from now on you know which buttons to press.

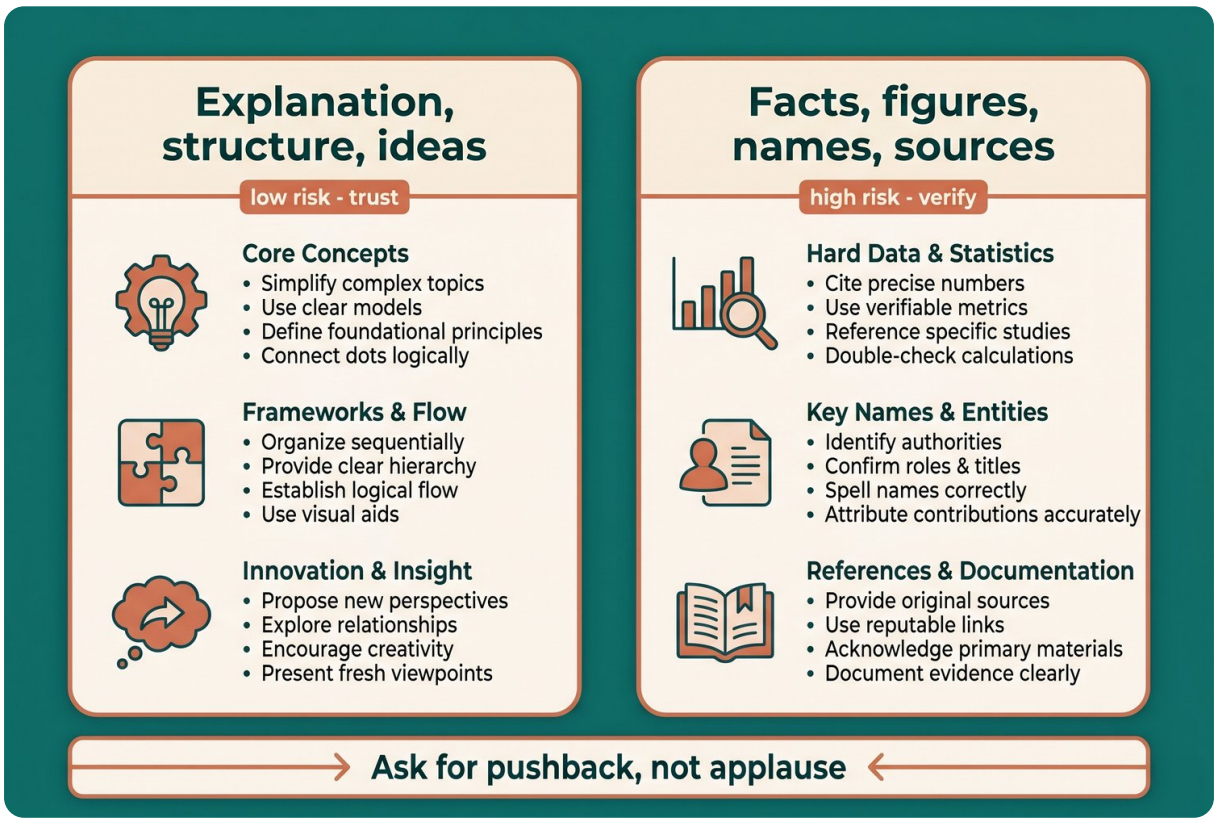


Figure - a split

In the next chapter comes the most important lesson of this book: it is not your prompt that determines quality, but the context you provide.

The big misconception: context beats the prompt

Almost everyone starts in the wrong place. They hunt for the perfect prompt: the magic sentence that suddenly makes Claude brilliant. That sentence does not exist. What does exist is context, and that is where almost all the quality comes from.

This is the most important chapter in the book. Remember only this, and you already get more out of Claude than most people. And the beautiful part: it is not a skill that takes years. It is a habit you pick up today.

The rule: context beats the prompt

Among people who work with AI daily this has become a cliche, and it is true: the teams that win with AI do not have a better model, they have better context. Researchers who lined up thousands of experiments reached the same conclusion. A mediocre prompt with good context beats a clever prompt with no context. Almost every time.

Why? Claude predicts the most likely answer based on what is in front of it (chapter 2). Give it nothing about your situation, and it fills the gaps with generalities, because that is statistically safest. Give it your documents, your examples, and your goal, and it does not have to guess at anything: it reasons about your reality. That is also why good context sharply reduces hallucination. With the real data in front of it, it does not have to invent anything; it can fall back on what you gave it.

Good context, concretely

Back to the quote from chapter 1. "Write a quote" produces something generic. But first give Claude your last three quotes, your price list, and your terms, and then lazily ask "turn this into a quote for this job." You get something in your style, with your prices, in your tone. The prompt got lazier, the result got better. That is the whole lesson in one example.

A few more, from other angles:

- A recruiter who asks "write a job posting" gets a template. Give it the role, two good old postings, and the company's culture, and you get something that speaks to the right people.
- A restaurant owner who asks "write a social post" gets something generic. Give it this week's menu, your tone, and an earlier post that did well, and it sounds like your place.
- A consultant who asks "summarize this meeting" with nothing more gets a neutral summary. Add what you are watching for and who it is for, and you get a summary you can actually forward.

Start with context, not with a question

The habit that seasoned users say pays off most is surprisingly simple: do not start a conversation with your request, but with three to five sentences of context. Who you are, who it is for, what you are trying to achieve. Then your question. The difference in quality is immediately noticeable, and it costs you ten seconds.

What Claude almost always needs:

- Who you are and what you do.
- Who the result is for.
- What the goal is, and in what tone.
- An example of good work, if you have one. One good example steers more than a paragraph of explanation.

Not more context, but the right context

A common overcorrection: dump in everything. A whole year of email, ten documents, a complete manual, while the question is something small. That backfires. Too much context that is not relevant distracts; the important things get buried (chapter 2, on the context window). Give the relevant context, not all context. For a quote: the price list and an example, not your entire bookkeeping. Quality and relevance beat volume.

Capture your context once

Here it gets powerful. You do not have to type that context again every time. You capture it once and reuse it. A short document with who you are, how you communicate, and how you want Claude to behave covers most of it. You put that in your profile settings (chapter 4), in a Project or in Cowork (chapter 10), and for recurring workflows in a Skill (chapter 11).

A second habit of power users: build your own library. When an approach works well, write it down. Not a list of internet tricks, but what demonstrably works in your work. After a few weeks you have a set of context blocks and instructions you reuse again and again, and that is exactly the difference between someone muddling along and someone for whom Claude really works.

Context first, then the question.

Build your context inventory (exercise)

Gather, once, the building blocks you need again and again. Half an hour of work that pays off for months:

- An "about us" text: what you do, for whom, what sets you apart.
- Your tone: three sentences that sound right, and three you would never say.
- Two or three examples of work you are proud of (a quote, an email, a post).
- Your standing rules: what must always happen, what may never happen.
- Your price list or rates, if they come up often.

Keep this in one place. This is the raw material for your settings, your Projects, and your Skills.

Pro tips from practice

- Start with context, end with your question. Three to five sentences up front change the whole answer.
- Give it your documents, do not ask it to recall them. Upload the source.
- Give relevant context, not all context. More is not better.
- Ask for the reasoning, correct, add. The gains are in the second and third round.
- One of the most underrated uses: have Claude explain something you already have. Paste a contract, an email, or a report and ask what it says.

Common mistakes and how to avoid them

- Hunting for the perfect prompt. Fix: stop wording, start giving context.
- Giving no background and then complaining it is generic. Fix: three sentences of context up front.
- Dumping in everything. Fix: give only what is relevant.
- Expecting it to remember your previous conversation. Fix: use a Project or memory (chapters 9 and 10).

Ready to use: your context block

Filled in here as an example. Paste it at the start of a conversation or save it as a project instruction (chapter 10), and replace the details with yours:

```
Who I am: Sofia, owner of Van Dijk Climate in the Netherlands, five technicians.
Who it's for: residential customers having heat pumps and AC installed.
How I communicate: plain and concrete, no sales fluff.
How I want you to work: think first, ask follow-up questions, be honest about
  uncertainty, push back where needed, and stick to my tone.
Goal of this conversation: write a clean quote for a new request.
Material: my price list and my best quote from last year (attached).
```

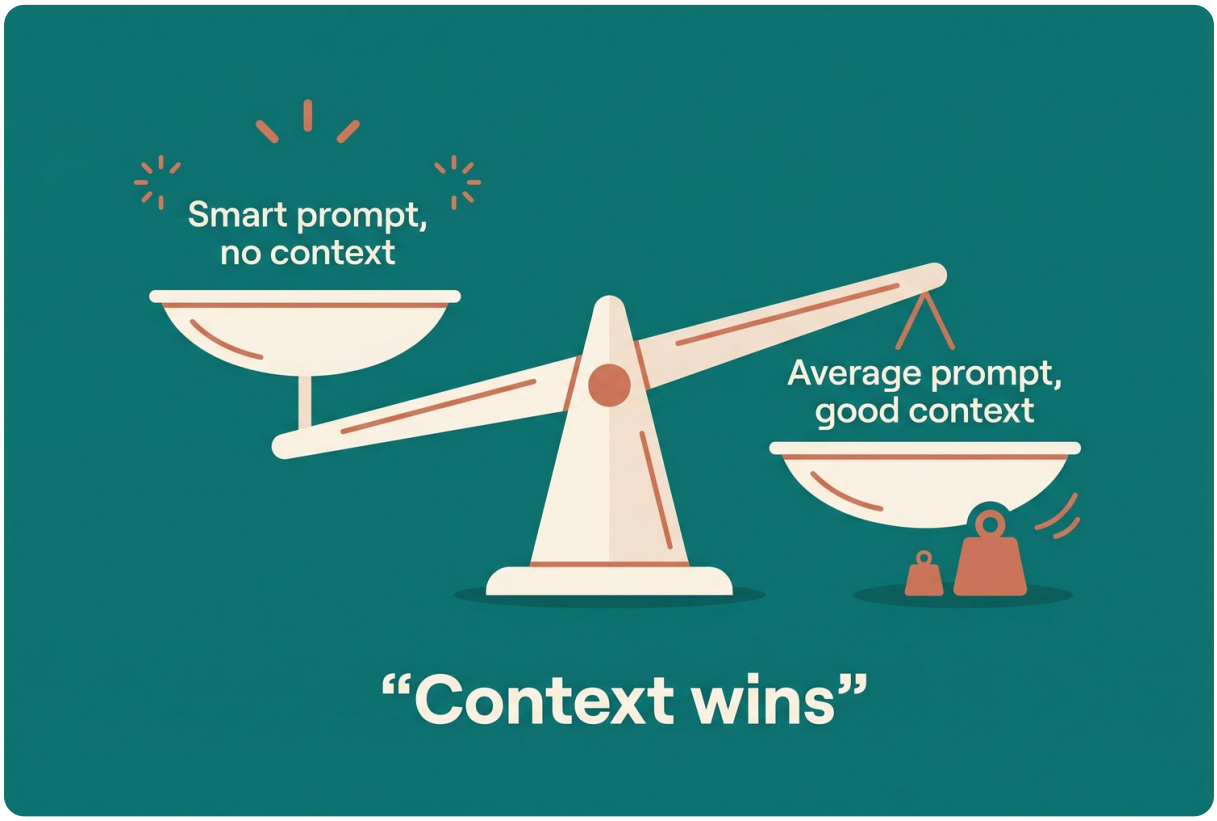
The lever shifts: from what you say to what you load

An insight seasoned users keep repeating: the techniques from a few years ago (endlessly tweaking the perfect sentence) now pay off less than what you load. The lever has shifted from the words of your question to the structure around it: your settings, your files, your examples, your memory.

That plays to a strength of Claude: it holds a lot of text at once. You can comfortably hand over a whole document, a few quotes, or a long email thread instead of summarizing it first. In short: stop polishing the sentence, and invest in the context you reuse every time. That is the cheapest and most durable quality gain there is.

Do this now

Build your context inventory today (the exercise above), and write your context block. Paste it at the start of your next three conversations and notice the difference. This one block is the cheapest quality gain there is, and everything in the parts that follow builds on it.



context wins. Clean, flat, one accent color.

Up to here it was about understanding. In Part 2 you get to work: starting in ten minutes, your first tasks with immediate results, and how to put this context lesson straight into practice.

Getting started in ten minutes

This chapter has two goals. First the practical one: within ten minutes you have an account, you know which plan fits you, and you have had Claude do one real task. Then the depth: how to set up your account right away so everything you do afterward gets better. Most people skip that second part and are still fighting the same noise months later. You do it right in ten minutes.

No technical knowledge required. If you can type a message, you can do this.

Step 1: account and app

Go to claude.ai and create a free account with your email, Google, or Apple. No credit card needed, no trial that quietly rolls over. Within a minute you are in the chat window.

Claude works everywhere and syncs automatically:

- In the browser on any device, at claude.ai.
- As an app on iPhone and Android, handy on the go and for dictating.
- As a desktop app on Mac and Windows, the nicest place for real work.

Start a conversation on your phone and finish it on your laptop in the evening. One account, the same conversation everywhere.

The desktop app has three tabs. It is good to keep them straight now, because you will hear the names again:

- Chat: an ordinary conversation. This is where everyone starts, and where most of your work happens.
- Cowork: a workspace where Claude carries out tasks on its own with your own files and folders. Chapter 10 is about this.
- Code: for those who build or automate software. Chapter 13 is about this.

You do not have to touch Cowork or Code now. Good to know they are there, so you know where to look later.

Step 2: set your context once, properly

This is the smartest first move, and it is chapter 3 put into practice. Instead of explaining who you are at the start of every conversation, you set it once in the settings, under what are called custom instructions or profile instructions. From that moment on, Claude carries that context into every new conversation, without you doing a thing.

Three things belong in it at a minimum:

- Who you are and what you do. For example: "I run an HVAC company in the Columbus, the Netherlands area, five technicians, mostly residential customers."
- How you communicate. For example: "Plain and concrete, no sales fluff."
- How you want Claude to work. For example: "Think briefly first, ask follow-up questions when something is missing, be honest about what you are not sure of, and do not invent facts or figures."

That is it. A few lines you never have to type again that bring every answer closer to usable. Later in this chapter there is a ready-to-use block you can copy. Later still, in a Project, you can add separate instructions per type of work (chapter 10), but this is the base that always travels with you.

Step 3: your first real task

Do not start with a test question like "tell me a joke." Start with work you had to do anyway, so you feel the gain right away. Give context, give the instruction, and give Claude the material it needs: drag a file into the window or type @filename to include it.

Five first tasks, from five angles, to get you going:

- HVAC contractor: "Write a friendly payment reminder for a customer who is two weeks late; mention the due date and keep it short."
- Cleaning company: "Here is a customer complaint I am pasting. Write a calm, solution-focused reply and propose one concrete next step."
- Online store: "Here is a product page. Write three shorter variants of the product copy, each with a different angle."
- Consultant: "Here is an hour of meeting notes as a PDF. Summarize it in one page: decisions, action items with an owner, and open questions."
- Restaurant: "Write a weekly-menu post for our social media in our tone; I will paste an example of an earlier post."

Read the answer and steer once: "shorter," "name the price," "a bit more formal." That steering is not a detour, it is the real work. The second version almost always lands.

Which plan fits you?

Check current prices on the site, because they change.

Plan	Price (approx.)	For whom
Free	€0	Trying it out and occasional use
Pro	around €20 per month (cheaper annually)	Daily use, all models, Projects, web search
Max	€100 to €200 per month	Heavy use all day, with priority
Team	around €30 per user, minimum 5	A business, shared workspace, commercial terms

What you get per plan, and when to scale up:

- Free is stronger than many people think. You get Sonnet (the fast, everyday model), web search, file uploads, Artifacts (Claude creates usable files), and memory across conversations. The limit is in the volume: roughly 15 to 40 messages per five hours. Enough to seriously sample, too little for all day.
- Pro, around €20 per month or €17 paid annually. Roughly five times the room, plus access to all models including Opus, unlimited Projects with persistent context, web search, and the more advanced environments (Claude Code and Cowork). For most people who work seriously every day, this is the right place.
- Max, €100 to €200 per month. Five to twenty times the room of Pro, with priority. For those who work intensively all day or run several agents at once.
- Team, around €30 per user (minimum five), or Enterprise for larger organizations. A shared workspace, administration, and more importantly, commercial terms: your data is not used to train models. That is exactly why a business with customer data belongs here, and not on a free personal account (chapter 7).

The rule of thumb from experienced users, and the most honest one: start free. Use Claude for a week or two. Keep in mind when the limit bothers you. If it bothers you several times a week during work that matters, Pro pays for itself in a few saved hours. If it rarely bothers you, just stay free. Do not pay upfront for a feeling; pay when you really hit the limit.

The three models, and when to use which

Remember the names, not the version numbers, because those shift every few months.

- Sonnet, the balance. Your default. Emails, quotes, summarizing, brainstorming, most analysis. Fast and strong enough for almost everything.
- Opus, the most powerful. For hard, long, or demanding work: a complex analysis, a long document that needs overview, a call you really want to get right.
- Haiku, the fastest and cheapest. For simple and high-volume: short tasks, quick translations, large numbers of small edits.

In doubt? Sonnet. Getting stuck on something complex, or getting a too-shallow answer? Switch to Opus and ask the same question again. You will notice the difference on exactly the work that matters.

There is also a mode for longer thinking (Extended Thinking). Turn it on and Claude visibly reasons step by step before it answers. Handy for analysis, tough choices, and multi-layered problems. For an email it is not needed; it only costs time.

Work clean: context hygiene

How you handle conversations partly determines quality. Three habits that make the difference:

- Have short, focused conversations with one goal, instead of one endless session where everything runs together.
- Start a new conversation for a new topic. Old context that is no longer relevant actually makes a new topic worse, because Claude keeps accounting for things that no longer matter.
- Give Claude the material instead of having it work from memory. Drag a file into the window or type @filename. An answer based on your real document is always better than an answer from memory.

Safety in one paragraph (the rest in chapter 7)

Two things already. One: Claude does not know your computer by default. Even in the desktop app you have to give explicit permission and point to files; that is by design, for your privacy. Two: do not paste sensitive customer data into a free personal account. If you work with customer data, set up a business version. Chapter 7 makes this complete; for now remember this one rule.

Live in two minutes.

Pro tips from practice

- Set your context once in the settings. The biggest quality gain for the least effort.
- Do not pay right away; let the limit tell you when Pro is worth it.
- Sonnet as default; Opus only when it stalls or stays too shallow.
- A new topic means a new conversation.
- Drag files in or use @filename; do not let Claude guess what is in a document.
- Speak instead of typing on your phone. For many people talking is faster than typing, and Claude turns it into clean text.
- Ask for the reasoning, not just the outcome, when it matters.
- Save what works. Keep a note with your best instructions; that becomes your own library.

Common mistakes and how to avoid them

- Getting Pro right away to be safe. Fix: start free and upgrade only when you hit a limit.
- Endlessly choosing between models. Fix: Sonnet, unless.
- Starting with a trivia question and concluding it is underwhelming. Fix: give a real task with context.
- Cramming everything into one endless conversation. Fix: a new conversation per topic.
- Typing your context every time. Fix: set it once in the settings.
- Describing a document instead of uploading it. Fix: drag it in or use @filename.

Ready to use: your standing settings

Set this once in the custom instructions in your profile settings:

```
Who I am: Sofia, owner of Van Dijk Climate in the Netherlands,
  five technicians, residential customers.
Who it's for: homeowners having heat pumps and AC installed.
Tone: plain and concrete, no sales fluff.
How to work: think briefly first, ask follow-up questions when something
  is missing, be honest about uncertainty, and do not invent facts or figures.
```

Ready to use: your first task

Paste this in a new conversation. Filled in here as an example; adapt it:

```
My task: write a friendly payment reminder for customer Johnson,
  whose invoice 2026-0418 is two weeks late.
Material: the invoice as a PDF (attached).
Form: a short email of at most eight sentences, with the due date in it.
```

Your first-session checklist

- Account created at claude.ai.
- Context block set in the settings.
- One real task done, with a file attached.
- Steered once to make the answer better.
- Web search tried on a current question (on a paid plan).
- Decided: free for now or Pro right away, and why.

Two things beginners do not know

- Check your usage. You can see how much of your limit you have used and when it resets (in the app, or with the /usage command in the advanced environments). Handy for deciding whether Pro is worth it: if you hit it often, the answer is yes.
- Start a prompt library from day one. Keep a note with your best instructions. Experienced users put it simply: keep a handful of reliable prompts and use at least three every week. That list becomes your biggest time-saver after a month, and later the base for your Projects and Skills.

Do this now

Create your account, set your context block once in the settings, and give Claude one real task from today with a file attached. Steer once. Done within ten minutes, and you not only have something usable in hand, you also have your account set up right for everything that comes next.

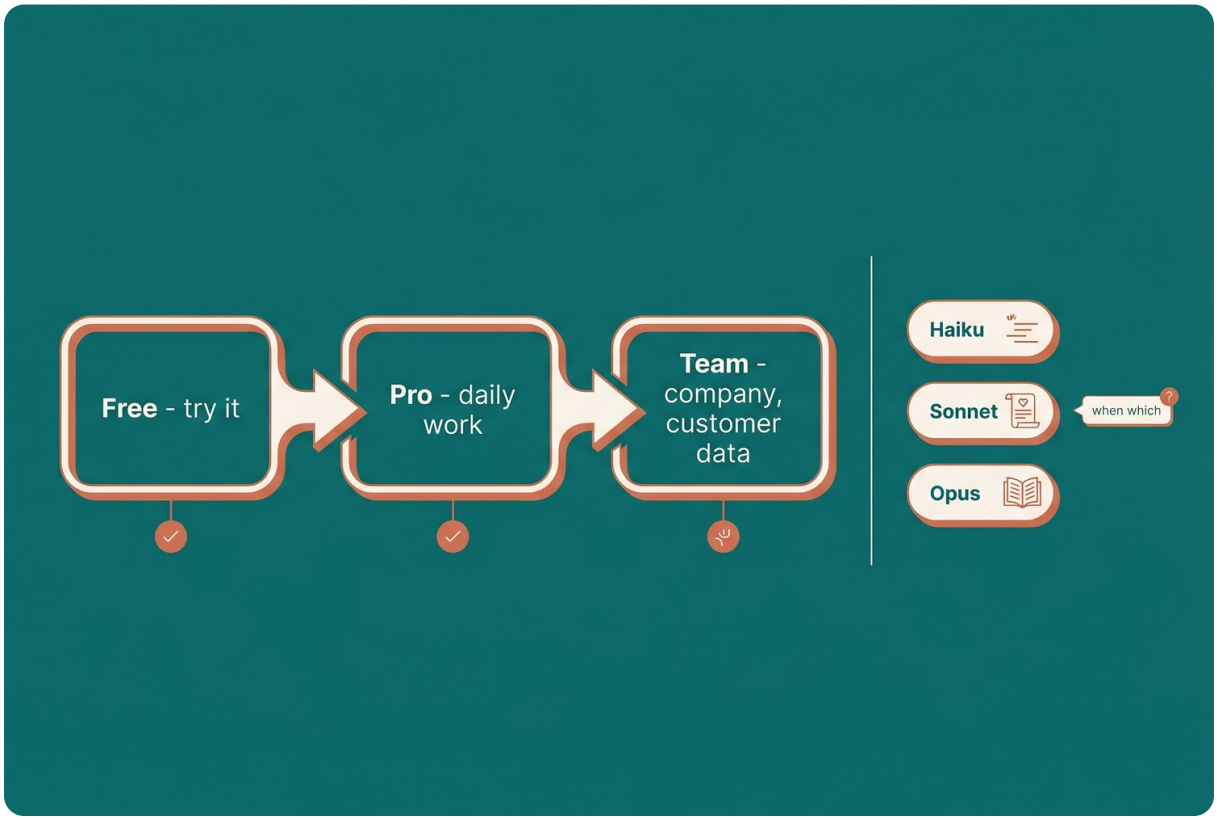


Figure - a simple decision aid

In the next chapter: ten tasks that save you hours today, each with a before and an after.

Your first ten tasks (with before and after)

The fastest way to understand what Claude does for you is to set it loose on ten real jobs. Not "what can AI do," but "what costs me time every week." This chapter gives you ten tasks that come up in almost every small business, each with a concrete example and an instruction you can copy. Do three of them this week, and you feel the difference in your calendar.

One thing up front, because it makes every task better. An instruction that works has four parts: the task (what needs to happen), the context (background and material), the form (how long, what format), and the constraints (what specifically not to do). That last one, saying what Claude should not do, sharpens the result more than any other part. Keep those four in mind; the examples below use them all.

1. Writing and answering email

The most everyday gain. Before: "Write an email to a customer" produces something bland and generic. After: "Write a short, friendly email to a customer who rescheduled his appointment twice; I want to confirm for next Tuesday, friendly but clear, at most eight sentences." Paste an incoming email and Claude writes a fitting reply on the spot that you only have to check.

2. Making a quote or proposal

This is where the big time gain is: a proposal that used to take you four to six hours takes twenty to thirty minutes of checking with good context. Give Claude your price list, your terms, and an earlier quote as an example, and describe the job. You get a quote back in your structure and tone. In chapter 11 you turn this into a standing Skill, so it is right every time.

3. Summarizing a long document

Paste or upload a report or a contract and ask for the core. This works well: "Summarize this in five points, name the key decisions, the deadlines, and the risks. Do not add anything that is not there." That last sentence matters: it keeps Claude from filling gaps with assumptions. For a contract, ask as a follow-up: "What obligations do I have, and what are the cancellation terms?"

4. Turning a meeting into actions

One of the most rewarding tasks. Paste your raw notes or a transcript and ask for a fixed structure. This saves the cleanup after every meeting, and the action-item table is usable right away. The ready-to-use instruction is later in this chapter.

5. Searching and querying documents

Upload a manual, a spec sheet, or a set of terms and ask questions about it as if you were asking a colleague who knows the document by heart. "Where does it say something about warranty?" "What are the requirements for completion?" You no longer have to scroll through eighty pages; you ask and verify the answer at the spot it points to.

6. Translating and adjusting the tone

The same text, a different coat. "Make this email more formal," "translate this into Spanish for a supplier," "write this more casually for our newsletter." Handy for anyone working with international customers or suppliers, and for anyone who has to get one message out across several channels.

7. Brainstorming without a blank screen

The blank screen is the biggest time sink. Claude fills it in seconds. "Give ten names for this service," "name ten objections a customer might have and how I counter them," "ten angles for a blog on this topic." You do not use it as the finished product, but as a starting point: you pick the two good ones and build on them.

8. Answering a complaint or tricky customer question

Paste the angry or tricky email and ask for a draft: "Write a calm, solution-focused reply. Acknowledge the inconvenience, do not get defensive, and propose one concrete next step. Do not promise anything about money or timelines." You keep control and only send after checking, but the first, hardest draft is there immediately.

9. Copy for social media or your website

Give Claude an example of a post that did well and your topic for today. "In this style, write a post about our new hours, at most a hundred words, plain tone, no exclamation marks." By giving your example, it sounds like your business, not like an ad flyer.

10. Making sense of a list or spreadsheet

Upload a CSV or Excel file with, say, your monthly revenue or your open invoices and ask what stands out. "What trends do you see, what jumps out, and what should I bring to my accountant?" Claude is strong at making sense of tables, but verify the numbers: have it show the reasoning and check the key figures yourself.

The boring prep work, in minutes.

The instruction structure that works

Remember the four parts and your instructions get better immediately. A filled-in example you can adapt for almost any task:

```
Task: write a reply to Mrs. de Vries's complaint below.
Context: I run a cleaning company; Mrs. de Vries complains that the team was
  late and skipped a window. I want to keep the customer. (Complaint attached.)
Form: an email of at most ten sentences, calm and solution-focused tone.
Constraints: do not get defensive, do not promise a refund, and do not add
  information I did not give.
```

Ready to use: meeting into actions

```
Here are my raw notes from a meeting (I paste them below).
Turn them into:
1. A short summary (3 to 5 sentences).
2. The key decisions (bullets).
3. Action items as a table: Task | Owner | Deadline.
4. Open questions that still need answering.

Important: do not add information that is not in my notes.
Notes:
Meeting June 3 with Anne and Mo. Decided: we take on De Bakker Bakery as a new
client. Anne does the quote by Friday, Mo handles the technician scheduling.
Still unclear: the electrical panel situation on site.
```

Pro tips from practice

- Start with a task you had to do anyway. You only feel the gain on real work.
- Use the four parts (task, context, form, constraints) and say explicitly what you do not want.
- Include an example of good work; that steers more than explanation.
- The first answer is a draft, not the finished product. Steer once or twice.
- Save the instructions that worked well; that becomes your own library (and later a Skill).

Common mistakes and how to avoid them

- Asking too vaguely ("make this better"). Fix: say what should be better and for whom.
- Giving no constraints. Fix: say what Claude should not do; that sharpens enormously.
- Sending the first answer right away. Fix: read, correct, follow up.
- Copying figures from a table on faith. Fix: ask for the reasoning and check the key numbers.

Non-obvious tasks that impress

The examples above are the starting point. This is where people are amazed, and where most of the time is to be won. Some of this works best with a connector or through Cowork (chapters 12 and 14), but the idea starts in a plain conversation:

- Overhaul the customer experience. Put your onboarding, your standard customer communication, and your follow-up into one Project and have Claude rewrite it into a tight sequence. This is where most owners leave the most time on the table.
- Invoice hunting. Have Claude pull your open invoices, assess each customer's payment behavior, and prepare a friendly draft reminder that you send with one click.
- Inbox triage. Have incoming mail labeled and the important messages surfaced, so you miss nothing in the noise.
- Review replies. Have polished responses to your Google reviews drafted, ready to approve. Good for your visibility, and you finally do it consistently.

The thread: these are no longer one-off questions, but recurring jobs you set up once and then let run. Exactly where the rest of this book is headed.

Do this now

Pick three tasks from this chapter that are on your list this week. Do them with Claude, using the four-part template. Note how much time each one saved you. That list is your own, honest business case for the rest of this book.

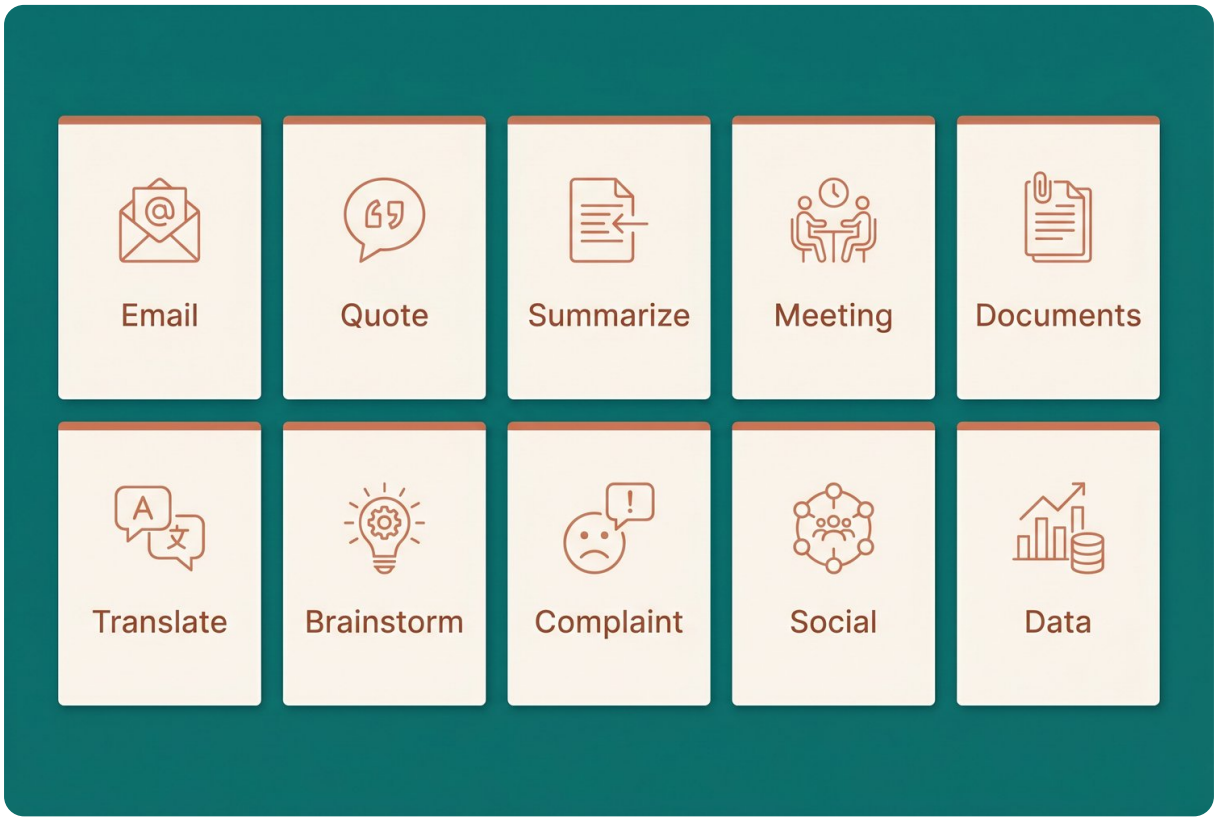


Figure - a grid of ten tasks (email, quote, summarize, meeting, documents, translate, brainstorm, c

In the next chapter we go from individual tasks to skill: how to learn to prompt so you land it in one or two turns on every task.

Learning to prompt well (in practice)

In chapter 3 you learned the most important lesson: context beats the prompt. This chapter is the layer on top of that. The few habits that let you land it in one or two turns on almost any task. No magic words, no tricks that stop working next month. A handful of habits you do automatically after a week.

Studies of knowledge workers find that people who work with a fixed, structured approach finish tasks noticeably faster than people who just wing it. The difference is not talent. It is structure.

The anatomy of a good instruction

You saw the four parts in chapter 5: task, context, form, and constraints. It is worth getting them sharp here, because almost every good instruction has them.

Weak: "Write a job posting." Strong: "Task: write a job posting for a technician. Context: HVAC company in the Netherlands, five people, down-to-earth culture, two years of experience required. Form: at most 200 words, with three reasons to work for us. Constraints: no cliches like dynamic or market leader, no list of ten job requirements."

The same minute of work, a completely different result. The first is a guess, the second is your posting.

Give it a role

Tell Claude what role to think from. "You are an experienced buyer, assess this quote critically" produces something different from a neutral read. "Think like a skeptical customer" surfaces the weak spots in your text. A role steers tone, depth, and angle in one sentence, more than a row of adjectives.

Show it, do not tell it

The most powerful way to convey your tone is an example. Paste an email or post that was good and say: "write in this style." One good example steers more than a paragraph explaining how you want to sound. This is called giving examples, and it is the difference between "it sort of resembles it" and "this is me."

Say what you do not want

This is the most underrated lever. Telling Claude what not to do sharpens the result more than any other part. "No sales fluff," "do not add assumptions," "no longer than a hundred words," "no buzzwords." Constraints give Claude edges to work within, and you a result you no longer have to clean up.

Ask for a form

Say what format you want it in. "In five bullets," "as a table with columns Task, Owner, Deadline," "an email of eight sentences," "a summary first, then the details." You get something you can use right away, instead of a slab of text you still have to knead.

Treat the first answer as a draft

The first answer is a rough version, not a finished product. The gains are in the second and third round. Steer with concrete directions: "shorter," "more formal," "name the price," "drop the intro," "give three variants." Sometimes this feels like a hassle, but it is faster than writing it yourself, and the result gets better every round.

One task per prompt

Give Claude five instructions in one message and you get five mediocre answers. Give it one and it gets your full attention. Cut big jobs into steps. First the structure, then the text, then the tone. That produces better work than asking for everything at once.

Let Claude improve your instruction

The most powerful trick for anyone still unsure about prompting: let Claude do the work. Say "Here is what I want to achieve. First ask me the questions you need to do this well, before you begin." Or: "Improve my instruction below and explain what you changed." You have Claude teach you to prompt, on your own tasks. Within a week you have it down.

Ask for pushback, not applause

From chapter 2: Claude tends to tell you you are right. If you want an honest judgment, ask for the opposite. "What is wrong with this plan?", "Name three risks," "Play devil's advocate." You get more useful answers than with "is this good?".

The second version is almost always the good one.

Pro tips from practice

- Use the four parts: task, context, form, constraints. Miss one and you notice it in the answer.
- Paste an example instead of describing your style.
- Be specific about what you do not want; that is where most of the gain is.
- Iterate out loud: just say what should be better, as if steering a colleague.
- Have Claude improve your prompt or ask you the right questions when you get stuck.
- Save your best instructions. What works for your work, works again tomorrow.

Common mistakes and how to avoid them

- Too vague ("make this better"). Fix: say what should be better, for whom, and in what form.
- Giving no constraints. Fix: explicitly say what you do not want.
- Asking five things at once. Fix: one task per prompt.
- Using the first answer right away. Fix: steer once or twice.
- Describing your style instead of showing it. Fix: paste an example.

Ready to use: a structured instruction

```
Task: write a job posting for an experienced technician.
Role: you are a compelling but down-to-earth copywriter.
Context: Van Dijk Climate in the Netherlands, five people, down-to-earth culture,
  two years of experience required; I attach a good old posting as an example.
Form: at most 200 words, with three concrete reasons to work for us.
Constraints: no cliches like "dynamic" or "market leader", and no list of
  ten job requirements.
```

Ready to use: have Claude sharpen your instruction

```
I want to achieve the following: a series of three emails to win back old
customers who have not ordered in a year, without sounding pushy.
First ask me the questions you need to do this really well.
Ask them one at a time, and only begin when you know enough.
```

The tricks of power users

A few patterns experienced users name again and again, and that you will not find in any manual:

- Force it to think before it answers. "Do not give a solution yet. First tell me what I am probably assuming wrongly, then ask me one question." This surfaces the mistaken assumption before you waste time.
- Make your role hyper-specific. Not "a buyer," but "an experienced buyer with fifteen years of experience, skeptical of overpriced quotes." The more specific the role, the sharper the answer.
- Bound the output. "Give me the 20 percent of actions that produce 80 percent of the result" prevents a list of fifty things of which three matter.
- Agree on standing codes with yourself. Some users put a short instruction up front like "skip the polite intro, give the answer directly." That way you train your own shortcuts.

A ready-to-use block, filled in here:

```
Do not give a solution yet. First tell me:
1. What am I probably assuming that is not true?
2. What information are you still missing to do this well?
Then ask me one focused question and wait for my answer.
```

Do this now

Take a vague instruction you gave Claude this week that fell short. Rebuild it with the four parts: task, context, form, constraints. Ask the same question again and put the two answers side by side. That difference is your new standard.

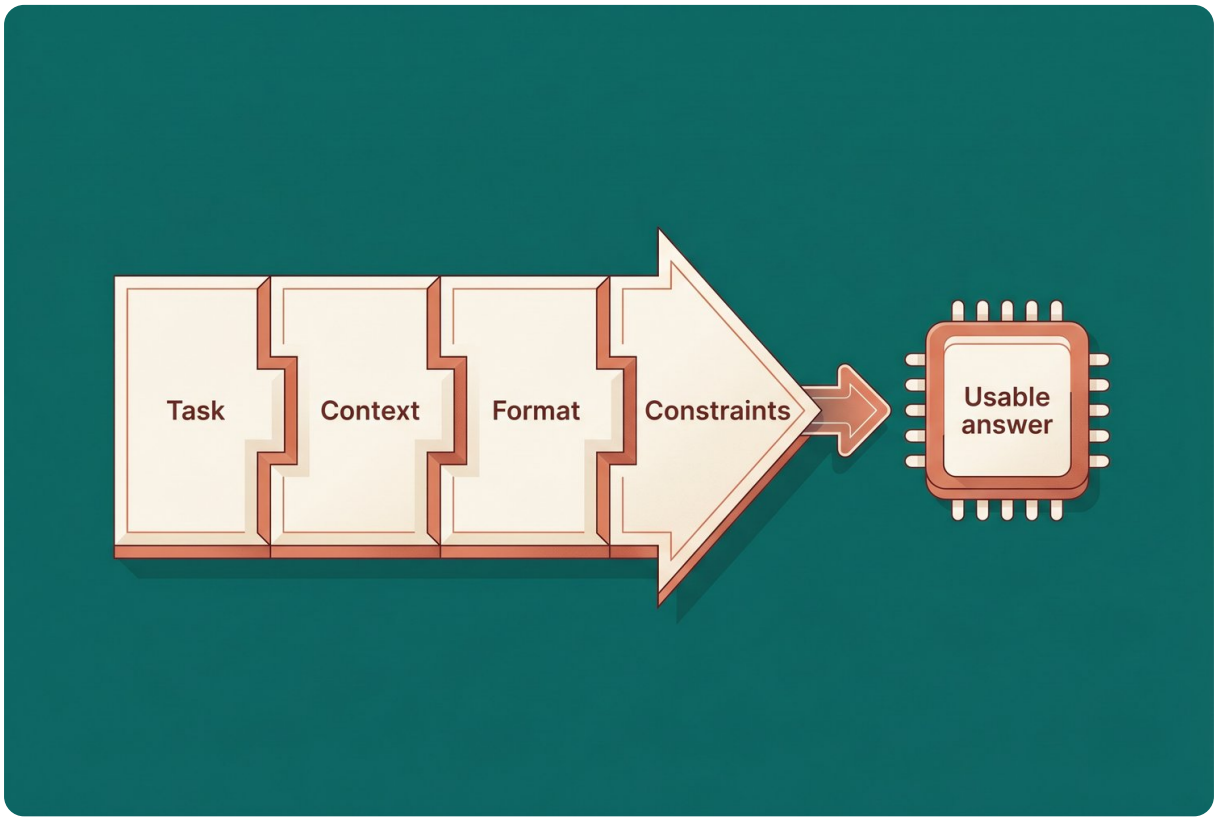


Figure - the four parts of a good instruction as four blocks (Task, Context, Form, Constraints) tha

Up to here it was about working well and fast. In Part 3 comes the question every owner asks first: how do I do this safely, especially with customer data?

Handling (customer) data safely

This is the question owners ask most, and rightly so: does the company behind Claude use my data? The worry is valid, but it is aimed at the wrong place. The answer does not depend on the brand, it depends on the plan you choose. Understand that one distinction and you use AI without data risk. Miss it, and you make the mistake most businesses make first.

This chapter is a bit longer, because it matters. As an owner, read all of it; as a user, at least the first half.

The core rule: consumer versus business

Claude comes in two worlds, and the difference is in the terms.

The consumer plans are Free, Pro, and Max. Since a change in August 2025, the rule there is: your conversations can be used to train the models, unless you turn that off yourself. The setting is on by default, behind a big "Accept" button, and data can be retained for up to five years if you do nothing. That is not malicious, but it is something you have to make an active choice about.

The business plans are Team, Enterprise, Claude for Work, and the API. There the opposite is arranged: training on your data is contractually excluded. Those business terms come with a Data Processing Addendum (a DPA), the kind of commercial data agreement you want when handling other people's information. For the API, data is not used for training and logs are kept only briefly. For those who truly need it, larger Enterprise customers can get an option with no data retention.

The one-sentence summary: if you work with customer data, you belong in the business world, not on a free personal account.

What you do concretely

- If you use a consumer plan, turn off the training setting today in your privacy settings. You can do this anytime, and it applies to future training.
- If you work with customer data or sensitive business information, set up a business plan (Team or higher) and sign the data processing agreement. Then your data terms are in order.
- Never paste passwords, national ID numbers (such as a BSN), complete customer files, or sensitive records such as medical or financial files into a consumer plan.
- Work by the principle: when in doubt, do not paste. Ask whether something is sensitive before you share it, not after.

The real risk is not in the model

Most AI data leaks do not happen because a model does something. They happen because an employee quickly pastes something into a personal free account, because the company never set up a business license. That is called shadow use, and it is predictable: if you do not give people a safe path, they take the easy one. The fix is as simple as it is effective. Set up a business license, and agree on one rule: sensitive data only in the business environment. Then no one has a reason to color outside the lines.

Not the model. The agreement.

The rules, in plain language (GDPR and the EU AI Act)

You do not have to become a lawyer, but a few things are good to know.

In the EU, the GDPR sets the bar for personal data, and it applies to every business that handles it. The core expectations are simple: you need a lawful basis to process personal data, you collect only what you need (data minimization), people have rights over their own data (access, correction, erasure), and you do not hand personal data to a third party who might reuse it for their own ends. A consumer AI account that may train on your input is exactly that kind of reuse, which is why customer data does not belong there. The business terms fix this: they come with a Data Processing Addendum, which under the GDPR makes the provider your data processor (Article 28) and contractually bars training on your data. Special-category data, such as health information, needs extra care and certainly never goes into a consumer account.

Then there is the EU AI Act, the first broad AI law, and two parts already matter for an SME. Since February 2025 there is an AI-literacy duty (Article 4): anyone working with AI should understand, at a basic level, what it does and where the risks are. This book plus a short internal briefing covers that. And from 2 August 2026 the transparency rules (Article 50) apply: tell people when they are dealing with an AI system, and label AI-generated content such as deepfakes. Most small-business uses are low or limited risk; high-risk uses, like AI in hiring or credit decisions, carry heavier duties but are rare. Chapter 8 turns this into a simple, low-cost plan.

(Selling into the US instead? There the picture is a patchwork of state privacy laws and FTC rules rather than one GDPR; the US edition of this handbook covers that.)

Keep a simple AI inventory

An AI inventory is just a short list: what you use AI for, with which data, and who checks it. For most small businesses it is not a standalone legal requirement, but it is the practical way to make oversight and transparency demonstrable, and it takes ten minutes. Chapter 8 turns this into a concrete one-page AI policy.

Pro tips from practice

- For customer data, always choose a business plan. The twenty dollars' difference does not weigh against one data leak.
- Turn off the training setting on every consumer account floating around your business.
- De-identify sensitive data if you still want to run something by it: replace names with "customer A."
- Tell customers honestly that you use AI for drafts and that a human checks everything. That builds trust and fits the transparency expectation.
- Make it easy for your people to do the right thing; banning without an alternative leads to shadow use.

Common mistakes and how to avoid them

- Pasting customer data into a free personal account. Fix: business plan, and the training setting off.
- Thinking the brand decides whether you are safe. Fix: the plan decides, not the brand.
- Using AI toward customers quietly. Fix: be transparent; it also builds trust.
- A strict ban with no alternative. Fix: give a safe, business path.

Ready to use: AI agreements for your team

Filled in here as an example for a fictional company; replace the name and the rules with yours:

```
AI agreements at Van Dijk Climate (version June 2026)

1. For customer work we use only the business Team plan of Claude.
2. No customer data, passwords, or national ID numbers (BSN) in a personal account.
3. In doubt whether something is sensitive: ask first, do not paste.
4. When using AI in customer contact, we disclose that we use AI; a human checks everything.
5. We keep a simple inventory of what we use AI for.
6. Point of contact for questions: Sofia.
```

Before you paste: a quick check

- Are there names, addresses, or account numbers in it? De-identify them or use the business plan.
- Is it sensitive data (health, finances)? Not in a consumer plan.
- Is this account a business one with training off? If not, set that up first.

What "opt-out" does not mean

A nuance few people know, and worth getting straight. Since late September 2025, Free, Pro, and Max fall under the consumer terms, with training on by default. But even with training off, a consumer account is still not a fully private, contractually covered environment:

- There is no Data Processing Addendum (DPA) on a consumer account. For contractual coverage and protection of trade secrets you need the business terms or the API.
- Messages can pass through safety filters and be reviewed manually if misuse is suspected.
- With training on, retention runs up to five years; off, it is shorter.

In short: "opt-out" stops the training, but it does not turn a consumer account into a business environment. For truly confidential work (customer files, contracts, source code) the difference between consumer and business terms is bigger than the marketing suggests. A cautionary real-world story underlines it: in a rare case, a user was shown someone else's document. Rare, but the lesson is simple: do not put your crown jewels in a consumer account, and always check what comes out.

Do this now

Three things, fifteen minutes of work. One: turn off the training setting on your own account. Two: decide whether you will move to a business plan for customer work, and if so, set it in motion. Three: rewrite the five rules above for your business and share them with your team. With that, you have taken the biggest data risk off the table.

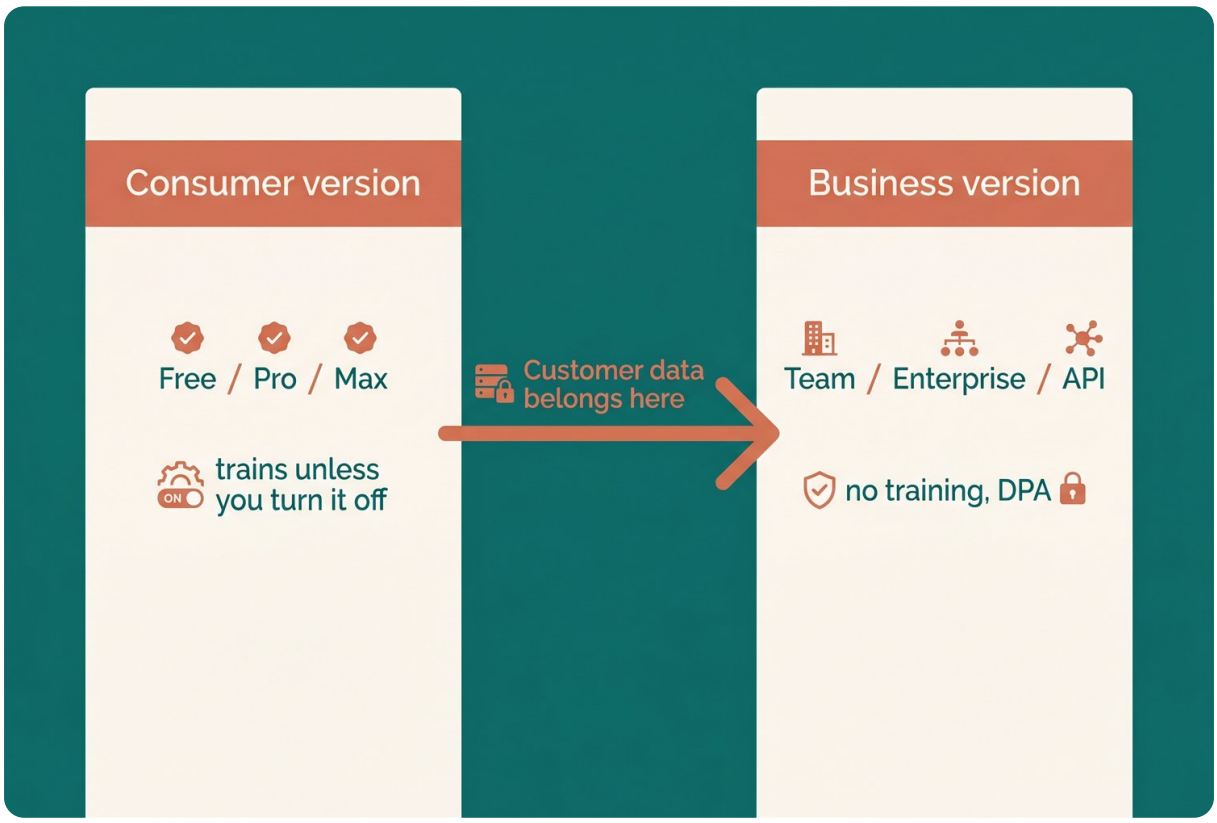


Figure - two columns

Rolling out AI in your business (light governance)

Here is an uncomfortable truth: someone in your business is probably already using AI, whether you have arranged it or not. The only question is whether it happens out in the open or in the shadows. This chapter is about that organization, and it can stay light. No thick policy document, no committee. One page, a few agreements, and one successful pilot.

That last part matters more than it sounds. Research has found that the vast majority of AI pilots fail, and almost never because of the technology. They fail on the approach: started too big, no owner, no measurement. This chapter keeps you away from those pitfalls.

Start with a one-page AI policy

A good AI policy for a small business fits on one page and answers six questions: what we use AI for, which tools are approved, which data may never go in, what is allowed and what is not, who checks, and who is the point of contact. You need no more than that to begin, and you sharpen it as you learn.

The most powerful part is the green, yellow, and red split: what is fine without thinking, what is allowed with approval, and what never. That gives your people clarity without having to ask about every little thing.

Roles and permissions

Keep it simple. One point of contact for AI questions (often the owner in a small business). One person who manages the business licenses and creates new accounts. And the agreement that a new AI tool only goes into use after a short approval, so you do not unknowingly put customer data into an unknown service.

Keep an AI inventory you actually fill in

An AI inventory sounds heavier than it is. It is a short list of what you use AI for, with which data, in which plan, and who checks it. It costs you ten minutes and gives you two things: a grip on what is happening, and demonstrable transparency if a customer or the law asks for it (chapter 7). A filled-in example is below.

One page is enough to start.

A pilot that actually succeeds

The way to really roll out AI is not "everyone can use it now." It is one team, one process, one month. Here is how:

- Pick one process that comes up often and costs time, for example quotes or customer emails. Do not start with something that touches eight systems; that strands for sure.
- Measure how it goes now first: how much time does it cost per week? That is your baseline.
- Have one team do it with Claude for a month. Train them briefly, and change nothing else, so you know where the difference comes from.
- Name an internal champion: someone who is enthusiastic and helps colleagues. Pilots with no owner fizzle out.
- Measure the same numbers after a month. Time saved times the hourly rate is your hard return; less hassle and more enjoyment is the soft one.

If it works, expand to the next process. If it does not, you know why, and you wasted little.

Bring your people along

Make sure the people who work with AI understand what they are doing. For a small business that is not a week-long course. It is this book, half an hour of explanation, and the agreements on your one-pager. Do it when onboarding new people and repeat it once a year. In the EU this isn't just good practice: AI literacy is a formal expectation under the EU AI Act (Article 4), in force since February 2025.

Pro tips from practice

- Start small and narrow. One process, one team, one month. Success spreads on its own.
- Measure up front, otherwise you can prove nothing afterward. A rough estimate beats nothing.
- Change nothing else during the pilot, otherwise you do not know what worked.
- Give the pilot an owner. Without a champion it fizzles.
- Revise your one-pager when a tool changes its terms or when you start using something new.

Common mistakes and how to avoid them

- Starting too big. Fix: one process, not your whole business.
- A one-week pilot. Fix: give it a month to judge fairly.
- Shifting the goals halfway. Fix: set the numbers up front and leave them.
- No owner. Fix: name a champion.
- A ban with no alternative. Fix: give a safe, approved path (chapter 7).

Ready to use: your one-page AI policy

Filled in here as an example for a fictional company; replace the details with yours:

```
AI policy Van Dijk Climate (1 page, version June 2026)

Goal: use AI to work faster and better, safely and transparently.

Approved tools: Claude (business Team plan) for customer work.
New tools only after approval from Sofia.

Data rules: never passwords, national ID numbers (BSN), complete customer
files, or medical data. Customer data only in the business plan.

Green (go ahead): drafts for emails, quotes, copy; summarizing;
brainstorming; translating.
Yellow (check first): anything with sensitive or large amounts of customer data.
Red (never): personal data in a personal account; letting AI make a final
decision without a human.

Human checks: anything that goes to a customer is reviewed by a human.

Point of contact: Sofia. Reviewed: every six months, or when terms change.
```

Ready to use: your AI inventory

A filled-in example; adapt the rows to your processes:

Process	For what	Which data	Plan	Who checks
Quotes	Drafting	Job details, no loose contact info	Team	Sofia
Customer email	Drafts and replies	Customer name, business plan	Team	Anne
Social posts	Writing copy	No customer data	Free	Mo

Ready to use: your pilot on one page

```
Pilot: making quotes faster with Claude
Team: the two people who currently make quotes.
Process: quote from request to sent.
Baseline: now averages 3 hours per week per person.
Duration: one month, change nothing else.
Champion: Anne.
Measure afterward: time per week, and how many quotes went out on time.
Goal: at least a third time saved, equal or better quality.
```

The tier choice is a leadership decision

Which plan you use for which work is not an IT detail but a leadership call, and it belongs in your policy. Set it down: sensitive customer work always goes through the business plan with a data processing agreement (chapter 7); general, non-sensitive work may run on a consumer account as long as training is off. That way the trade-off does not land on an employee who cannot see the legal side, and everyone knows at once where each kind of work belongs.

Do this now

Two things. Rewrite the one-pager for your business today, with the green-yellow-red split. And pick one process for a one-month pilot, with a baseline and a champion. With that, you turn AI from loose experimentation into something you can steer and prove.

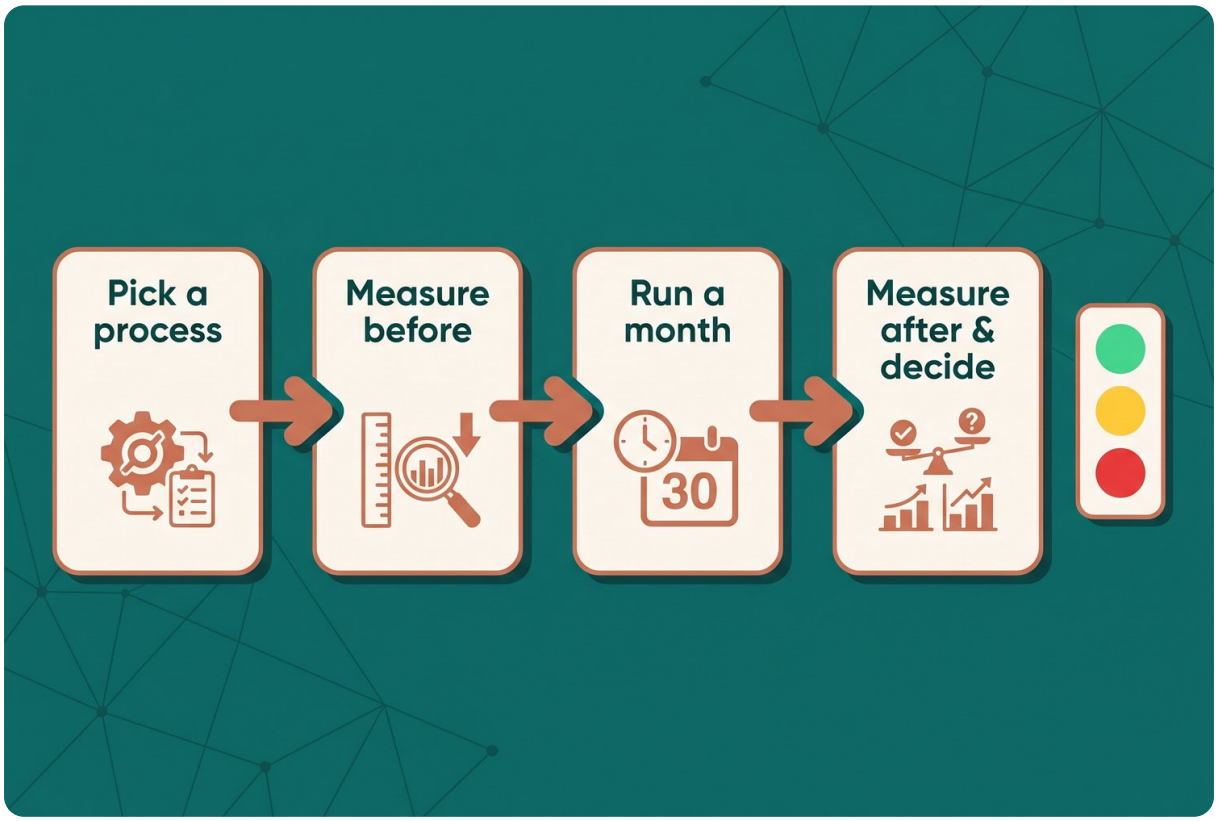


Figure - a four-step pilot timeline: pick a process -> measure first -> run for a month -> measure

Up to here it was about understanding, starting, and rolling out safely. In Part 4 we go deep: all of Claude's capabilities, from the basics (artifacts and projects) to the advanced (skills, scheduled tasks, Claude Code, and connectors), so AI turns from a handy assistant into something that truly knows y

Artifacts and files: have Claude create documents

Most people stay stuck in chatting: ask a question, read the answer, copy, paste. The bigger leap is having Claude make things and work with your own files. Then it stops being a conversation and starts delivering real results: a clean quote as a Word file, a price calculation as an Excel with working formulas, a presentation, a report with a chart. This is the base layer of what Claude can do, and since early 2026 file creation is in every plan, including the free one.

Claude creates real files for you

Ask in plain language for a document, and Claude delivers a file you can download right away or drop into Google Drive. Concretely:

- A quote or letter as a Word file, in your structure.
- An Excel with working formulas: a quote calculation, a timesheet, a simple cash-flow overview, including a small chart.
- A PowerPoint from your notes or a document.
- A PDF to send.

On top of that, Claude can build small, working things you see alongside the conversation: a calculator, a checklist page, a small overview. Those are called Artifacts. You do not have to install anything; you ask, and it appears. Files can be up to thirty megabytes, plenty for daily work.

Claude works with your own files

It goes the other way too, and that is maybe even more useful. Drag a file into the window and have Claude work with it. It reads PDFs, Word, Excel, CSV, and images. A few examples:

- Upload a contract and ask for the core, the obligations, and the cancellation terms.
- Upload a spreadsheet with your revenue and ask which trends stand out.
- Take a photo of a handwritten or scanned invoice and have the data pulled out; Claude can read images.
- Paste a long product list and have it turned into categories.

The chain: from PDF to presentation

The power is in combining. You pull tables from a PDF, have them cleaned up in an Excel with a summary chart, ask for a Word report with the conclusions, and turn that into a presentation. Four steps, one running conversation. Since May 2026, Claude also works as an add-in inside Excel, Word, PowerPoint, and Outlook, carrying your context from one program to the next without you having to paste anything again.

You ask for it, and it is a file.

Web search and memory, briefly

Two features you can already use here. Web search pulls current information on the paid plans, with a source attached; handy when you need something from today instead of from training. And memory lets Claude remember how you work across conversations; you can view, edit, or wipe those memories. In the next chapter we build that out into real projects and knowledge bases.

Concretely for a small business

- An HVAC contractor turns a dictated site note into a quote as a Word file, with a calculation as an Excel beside it.
- A bookkeeper has the tables from ten PDF invoices put into one Excel, with the overdue items flagged.
- A consultant turns a report into a ten-slide presentation, ready to refine.

Pro tips from practice

- Explicitly ask for a file: "make this into an Excel" or "deliver this as a Word document."
- Give the format you want: column names, number of slides, with or without a chart.
- Upload the original instead of retyping it; just drag it into the window.
- Check what Claude pulls from a document or photo; vision is strong but not infallible, especially with bad scans.
- Have it report what it could not read, so you know where to look yourself.

Common mistakes and how to avoid them

- Expecting Claude to know a file you did not upload. Fix: drag it in.
- Copying numbers from a scanned invoice on faith. Fix: have the source shown and check the amounts.
- Asking for a vague format and then having to format it yourself. Fix: name columns, length, and style up front.
- Uploading a low-quality scanned photo. Fix: take a straight, sharp photo or use a real scan.

Ready to use: have an Excel created

Filled in here as an example; replace the numbers with yours:

```
Create an Excel file "quote calculation".
Columns: Item, Quantity, Unit price, Subtotal (as a formula).
At the bottom: subtotal, VAT (7%), and grand total.
Fill it with this example: 2 AC units at €950, 40 feet of line set at
€6 per foot, 6 hours of labor at €85 per hour.
Add a small bar chart with the subtotal per item.
```

Ready to use: read a PDF into Excel

Filled in here as an example; adapt it to your file:

```
Here is a PDF with a list of invoices (attached).
Convert the table into an Excel with columns Invoice number, Customer, Amount,
Due date. Flag the invoices that are past due.
Do not add data that is not in the PDF, and report what you could not read.
```

Pro tips that lift your files to your brand

- Turn it on. In Settings, under Features, there is "enhanced file creation and analysis"; some business accounts have to enable it at the organization level first.
- Give it your brand. Upload your font names and your brand colors (in HEX), and Claude makes files that look like yours right away. Ask for SVG if you want to finish it in a design tool.
- Publish and share. You can publish an Artifact as a link others can view without an account, and with an account copy and adapt.
- Mind the sandbox. File creation runs in a walled-off environment with limited internet access to fetch helper files. Keep an eye on the intermediate steps and stop if something odd happens, and always check numbers and formatting; a conversion sometimes goes wrong.
- One limit to know: you cannot (yet) upload an existing PowerPoint as a house-style example. Give your style as separate instructions or a PDF instead.

Do this now

Take a document you often make by hand, for example a quote or an overview. Have Claude make it once as a real file, with the right columns or structure. Also upload a PDF you would normally dig through and ask three questions about it. You feel the difference between chatting and having it made right away.

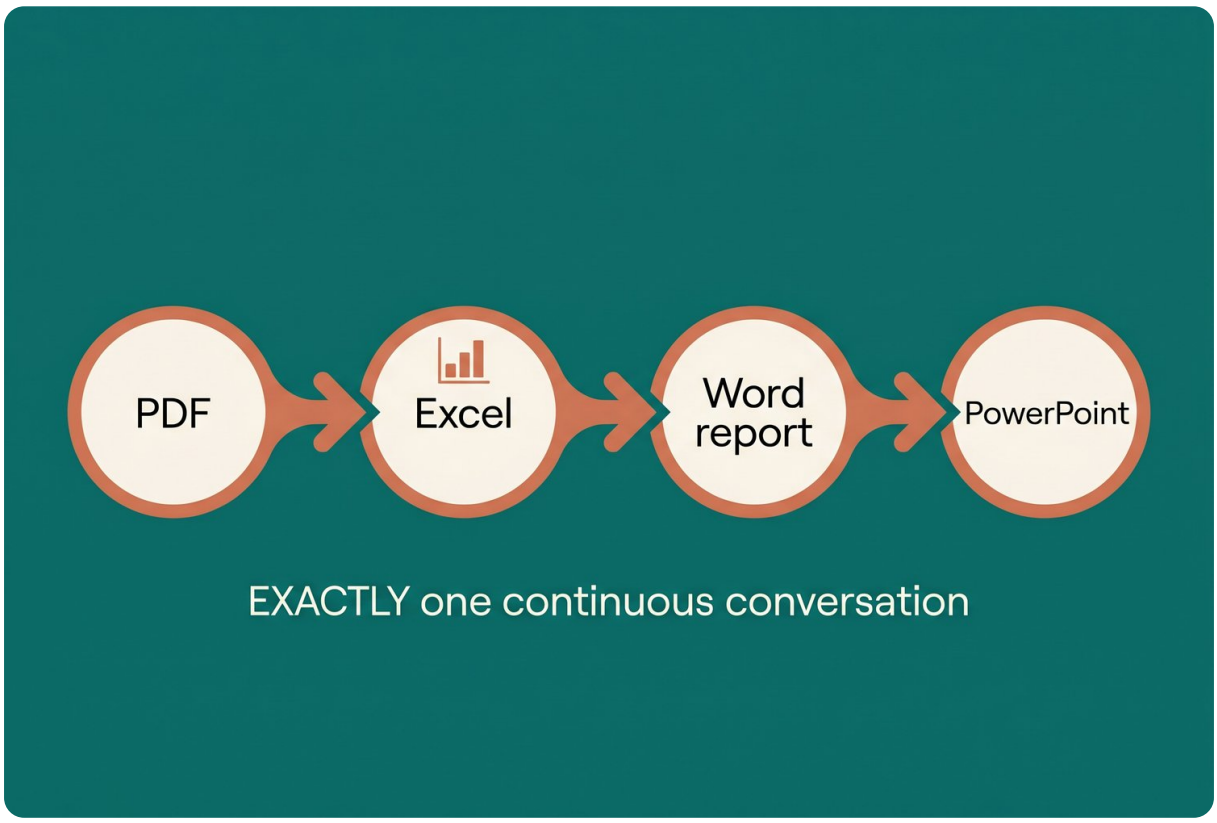


Figure - a chain of four blocks: PDF -> Excel (with chart) -> Word report -> PowerPoint, with an ar

In the next chapter we take the second rung of the ladder: projects and knowledge bases, with which you make context permanent and every conversation starts with your knowledge already in place.

Stop with loose chats: projects and knowledge bases

Go to the forum r/ClaudeAI and you see the same frustration come up again and again: "Why do I have to keep reminding Claude to look at my documents?" and "I explain everything from scratch every time." That is not a fault in Claude. It is the difference between someone who has loose chats and someone who works in a Project. This chapter is the second rung of the ladder, and for most small businesses this is where the real gain is.

A loose conversation is disposable. A Project is a workspace that knows your business, every time, without you repeating a thing.

The three building blocks of a Project

A long-time user summed it up nicely on the forum. A Project has three parts:

- Project knowledge: your knowledge base. All the facts, documents, and details that matter. Your services, prices, terms, your best examples, your SOPs. Claude uses this as background in every conversation in the Project.
- Project instructions: how Claude should think and answer. Your role, your tone, your rules. It is your context block from chapter 3, but captured for this entire Project.
- Chats: the loose conversations. Think of them as specialists who all know the same background but each take a different job. One writes a quote, another answers a complaint, both know who you are.

The nice part: a conversation you lose in an ordinary chat once you close it stays saved in a Project, including the uploaded files. Projects are available to everyone, including free (up to five). On a paid plan the knowledge base scales automatically: if you add a lot of documents, Claude switches to smart retrieval (RAG) and expands capacity roughly tenfold, without quality sagging.

What belongs in your knowledge base

Start with what experienced users upload first: a company overview. Who you are, what you do, for whom, and what your goals are for the coming quarter. Then you build out with what recurs in your work:

- Your services, prices, and terms.
- Your tone: a few texts that sounded right, and a few you would never say.
- Your best examples: a strong quote, a good customer email, a clean work order.
- Your SOPs and standard ways of working.
- Frequently asked customer questions with your standard answers.
- Your jargon, explained.

And just as important: what does not belong in it. No outdated documents (Claude does not know which version is the right one), no loose clutter that is not relevant, and in a consumer plan no sensitive customer data (chapter 7). Quality and relevance beat volume; a knowledge base full of noise actually makes the answers worse.

The problem nobody tells you about (and the fix)

Here is the most-shared complaint from power users, and the fix right with it. Claude does not always consult your project knowledge on its own. It can give an answer that is right in form while ignoring the upload. On the forum someone describes exactly this: Claude invents a method, you point it back to the documentation, it apologizes and then does it right. Endless repetition.

The fix is in your project instructions. Put in explicitly:

- "Always consult the project knowledge first before you answer."
- "If you are unsure about something or it is not in the knowledge, say so and do not make anything up."
- "Reference the document your answer is based on."

Those three rules remove most of the frustration. It is the same principle as in chapter 2: you have to tell Claude how to behave, otherwise it takes the easiest route.

A workspace that knows your business.

A power-user trick: make your knowledge live

By default, project knowledge is static: Claude reads it but does not update it. For anyone using a Project to manage ongoing work that is awkward, because you are constantly replacing documents. Advanced users solve this with a living document: a text file you have updated through the desktop (Cowork) or a connector, so the Project keeps itself current. A variant going around the forums: a main Project with the central knowledge, and sub-Projects that reference it, so you effectively build a virtual team that shares the same, continuously updated knowledge. This is not an official feature but a clever workaround; good to know it is possible once you are ready.

Cowork: your Project as a real workspace

On the desktop this goes a step further with Cowork. There a Project is a real folder on your computer, with its own context files, working files, scheduled tasks, and its own conversation history. Three things experienced users emphasize:

- Describe the outcome, not the steps. "Turn each of these six transcripts into a separate summary with action items" works better than a long list of micro-instructions. Cowork divides it over parallel subtasks on its own.
- Tell Claude what context it works in. A sentence like "you are working in a Cowork session with access to the PROJECTS folder, save your output in the OUTPUT folder" prevents a whole class of mistakes.
- Give each area of work its own Project. One Project per client or per workflow keeps context from running together.

And the most powerful part: scheduled tasks. You describe a recurring job once and Cowork runs it automatically, daily or weekly. A morning briefing that summarizes your email and calendar, a weekly report from your spreadsheets, a monthly competitor scan. You can even queue tasks from your phone and have your computer run them overnight. One important condition: your computer has to stay on with the app open, otherwise the scheduled task does not run.

The non-obvious examples

- A knowledge base that lets a new hire perform at a senior level. Put your best quotes, your tone, and your SOPs into a Project. A new hire opens the Project and delivers work from day one that sounds like your most experienced person, because the knowledge is not in someone's head but in the Project.
- A quote process that orchestrates itself. One Project with your price list, terms, and examples; you paste a dictated site note and get a complete quote plus a draft follow-up email in your style.
- A content system. Upload the transcripts of your videos or your expertise, and have blogs, posts, and newsletters pulled from them that actually use your content, not generic talk.
- A daily briefing via Cowork. Every morning at seven, Claude summarizes your inbox, calendar, and a news source and sets it ready, before you have your first coffee.

Pro tips from practice

- Put in your project instructions that Claude always consults the project knowledge first and invents nothing. This is the most important tip of the whole chapter.
- One Project per client or workflow. Not everything on one pile.
- Describe the desired outcome, not every step.
- Tell Claude explicitly which environment it works in (ordinary chat, Project, or Cowork session).
- Keep your instructions short and your knowledge base clean; remove outdated documents right away.
- Add your best example; one excellent example steers more than ten lines of explanation.

Common mistakes and how to avoid them

- Dumping everything into one mega-Project. Fix: split per client or per type of work.
- Stuffing the knowledge base with old or irrelevant documents. Fix: only what is correct and relevant; clean up.
- Not saying that Claude should use the knowledge. Fix: put it explicitly in the instructions.
- Never updating the knowledge base. Fix: schedule a fixed moment (see below).
- Sensitive customer data in a consumer Project. Fix: business plan (chapter 7).

Ready to use: your project instructions

Filled in here as an example; replace the details with yours:

```
You work in the Project of Van Dijk Climate (the Netherlands, five technicians,
residential customers). Tone: plain and concrete, no sales fluff.

How to work:
- ALWAYS consult the project knowledge first (price list, terms, examples)
  before you answer.
- If something is not in the knowledge or you are unsure, say so and invent
  nothing; ask follow-up questions if needed.
- Briefly reference the document your answer is based on.
- Stick to our tone and our standard structure for quotes and emails.
```

Ready to use: your knowledge-base structure

A filled-in example; adapt the files to your business:

```
Project knowledge Van Dijk Climate:
- 00-company-overview (what we do, for whom, goals this quarter)
- 01-price-list-2026
- 02-terms-and-conditions
- 03-tone-and-style (what to do, what not)
- 04-example-quote (our best from last year)
- 05-example-customer-email
- 06-faq-with-answers
- 07-completion-process (SOP)
```

Ready to use: your maintenance rhythm

```
- Every month: check the price list and terms, replace outdated docs.
- After every good output: add it as a new example in the knowledge base.
- Every quarter: update the company overview and goals.
```

One rung deeper

Two things advanced users share. One: many people keep their real way of working in loose files (for example in a notes app) and have Claude run through them with a single instruction, instead of fighting a rigid setup; you simply edit your file. Two: a main Project with your central company knowledge works as a kind of company brain that you have other Projects reference, so you do not have to update the same things in ten places.

Do this now

Create one Project today for the work you do most. Put your context block from chapter 3 into the project instructions and add the rule that Claude always consults the project knowledge first. Upload three things: a company overview, your price list or terms, and your best example. Then run a real job in it. You notice at once that you no longer have to explain anything.

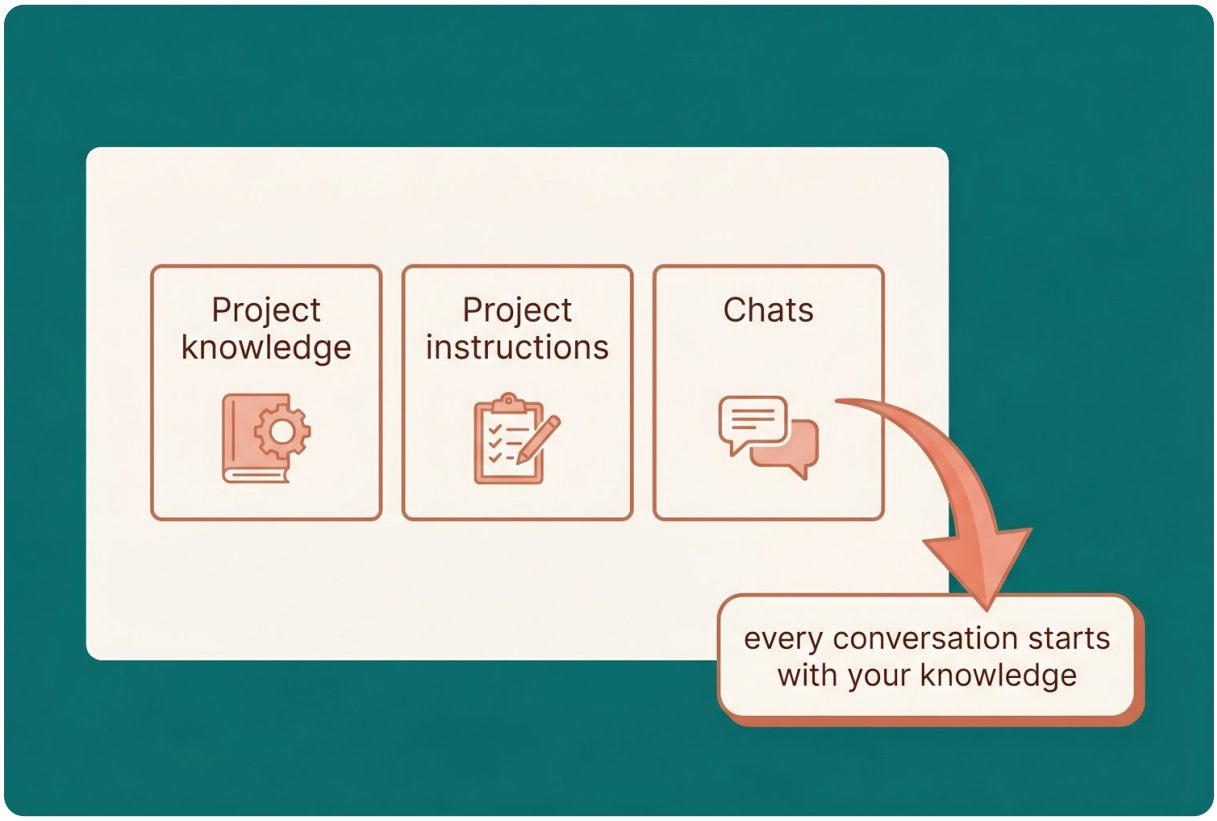


Figure - a Project as a workspace with three building blocks (Project knowledge, Project instructio

In the next chapter we take the third rung: Skills, with which you capture recurring work as a capability that Claude applies on its own.

Skills: recurring work becomes a capability

Claude has a feature that, according to experienced users, ninety percent of people do not know about, and that is maybe more powerful than any connector. Skills. And the best summary going around the forums nails it: a prompt is a request, a Skill is a job description.

You know the problem they solve by now. Conversation after conversation you explain the same things: write in this tone, use this structure, follow these rules. With a Skill you capture that once and Claude pulls it in on its own, in every conversation, forever. This is the third rung of the ladder, and once you get used to it, working without Skills feels like onboarding your colleague again every morning.

What a Skill actually is

A Skill is, at its core, a folder with one text file: SKILL.md. At the top are two things, a name and a description, and that description is the trigger: it is how Claude recognizes when to apply the Skill. Below that are your instructions. That is all that is needed, and you make one in five minutes.

The cleverness is in something called progressive disclosure, tiered loading. Claude keeps only the name and description of each of your Skills on hand (a few dozen words each). Only when a task fits does it read the full instructions, and any extra files only when needed. That lets you have a large library of Skills without slowing down your conversation. You capture a lot, Claude uses only what fits.

Skill, prompt, or connector?

Three things get confused often:

- A prompt is one-off. You type it, you get an answer, and next time you start over.
- A Skill is a job description that stays in place and loads on its own. Role, steps, rules, examples.
- A connector (MCP, chapter 14) links Claude to your data and systems.

The best line from the community: a connector gives Claude access to your data, a Skill teaches Claude what to do with it. They reinforce each other. A connector without a Skill is powerful but generic; together you get a Claude that works like your best employee.

The standard Skills you already get

For common work there is already a set included. The best known is the document suite: it creates Word, Excel, PowerPoint, and PDF, and is built into the paid plans. Anthropic also shares a public catalog of example Skills (with tens of thousands of stars on GitHub), where you get a sense of what a Skill can be:

- Theme Factory and Brand Guidelines: upload your house style once, and every file Claude makes follows your colors and fonts.
- Systematic Debugging: have Claude work from cause to fix like an experienced developer.
- Webapp Testing, MCP Builder, Slack GIF Creator, and more.

There are even partner Skills from companies like Atlassian, Figma, Canva, Stripe, and Notion. Good to know this exists; for your business it gets truly powerful only when you make your own Skills.

Your way of working, captured once.

What makes a good Skill

Treat a Skill like a job description, not a prompt. That means: a clear role, hard rules, and the desired result. The best candidates are tasks you do often that have a fixed form: a quote, a customer email, a work order, a social post. The rule of thumb that comes up everywhere: if you explain something a second time, make it a Skill.

Two things decide whether it works. The description (the trigger) must be sharp, so Claude picks the Skill at the right moment. And there must be one excellent example in it, because that steers more than ten lines of explanation.

A Skill is a package, not a sticky note

Here most people make a mistake. The powerful Skills are not one little text, but a small package. Alongside the instruction file you can include folders:

- references: your documents Claude reads in, such as your price list, your terms, and your style guide.
- scripts: small pieces of code for tasks that need exact logic, for example a calculation or fixed formatting. You do not have to write these yourself.
- assets: templates, fonts, or logos that end up in the output.

Claude reads these in at the moment it uses the Skill. That way your quote Skill works with your current rates and your tone, not a guess. And maintenance is simple: change your prices and you update one reference file; the Skill stays the same.

Build your own, without programming

You need no technical knowledge, and it takes fifteen minutes. The easiest path runs through the skill-creator, a helper Skill that interviews you and builds the rest for you:

- Turn on Skills in the settings (under Capabilities), and turn on the skill-creator.
- Open a new conversation and say: "Use the skill-creator to make a Skill for my quotes."
- Describe your task in detail and be specific about the result. Claude asks a few clarifying questions.
- Claude builds the complete Skill file plus a read-me with your next steps.
- Read the read-me, download the file, and upload it in the settings under Skills. Flip the switch on.
- Test it in a new conversation by asking Claude to use your Skill by name.

If something does not work the way you want, go back to the conversation where you made it, say what should be better, and upload the new version. Start with something simple you already do often; you learn the rest as you go.

Advanced: keep them sharp, and let them improve themselves

Two things the real power users share. One: keep a Skill short and reference separate files, instead of cramming everything into one long file. A large comparative experiment found that a hybrid setup (a short core with references to detail files) does better than everything-in. Prune your Skills now and then; remove what no longer fits.

Two, and this is the nicest: a self-improving Skill. Advanced users make a meta-Skill that watches their work, notes where it can be better, and applies those improvements to the other Skills at a fixed moment. You effectively build a system that sharpens itself. You do not need this to start, but it shows how far it goes.

Non-obvious Skills that impress

- A review-reply Skill: monitors your Google reviews and prepares polished draft responses (good for your visibility).
- An invoice-hunt Skill: pulls your open invoices, assesses payment behavior per customer, and prepares friendly reminders.
- A quote Skill as a package, like the one below.

Pro tips from practice

- Write the description as a sharp trigger ("use for quotes"), not as a vague sentence.
- One Skill, one task. A Skill that tries to do everything does nothing well.
- Put your references in separate files; then you update your price list without touching the Skill.
- Let the skill-creator do the heavy lifting and iterate on a real example.
- Be careful with ready-made Skills from the internet; use them at your own risk and read what they do first.

Common mistakes and how to avoid them

- Cramming everything into one long SKILL.md. Fix: short core plus references.
- Too broad. Fix: one Skill per task.
- Including no example. Fix: put your best work in it.
- A vague trigger description. Fix: say concretely when it should load.
- Blindly trusting a community Skill. Fix: read it first, test in a new conversation.

Ready to use: a complete quote Skill

Filled in here as an example; replace the italic details with yours. Save the instruction part as SKILL.md in a folder "quote", with your own documents in a folder "references" beside it.

```
---
name: quote
description: Use this as soon as I ask for a quote, price estimate, or
  proposal for a customer, including from a short or dictated note.
---
# Making a quote

## When to use
For any request for a quote or proposal.

## Steps
1. Pull scope, size, location, and timeline from my input.
2. Calculate with the rates in references/price-list.md (tiered from 3 days).
3. Follow the structure and tone in references/style.md.
4. Add the right passages from references/terms.md.
5. Close with validity (30 days) and a concrete next step.

## Rules
- Never name a price that does not come from the price list.
- Do not give a discount without my approval.

## Example
Write in the style of references/example-quote.md.
```

In the bonus toolkit at the back of this book there are five more ready-to-use Skills (customer email, weekly report, social post, complaint handling) you copy the same way.

The skill library: take them, they are already made

You do not have to reinvent the wheel. The community has made thousands of Skills, free to use and to read before you install them. Below is a curated selection with what they do and where to find them, with credit to the makers. At the end is how to grab them and a safety note.

Where to find thousands of Skills

Collection	What is in it	Where to find it (credit)
Official Skills	Document suite, theme-factory, skill-creator and more	GitHub: anthropics/skills (official)
Awesome Claude Skills	1000+ practical, automation, and cross-tool skills	GitHub: ComposioHQ/awesome-claude-skills
Awesome Agent Skills	200+ hand-picked, from official teams	GitHub: VoltAgent/awesome-agent-skills
Antigravity Awesome Skills	1200+ skills, the largest collection	GitHub: sickn33/antigravity-awesome-skills
Curated lists	By category, with stars and date	GitHub: travisvn and BehiSecc/ awesome-claude-skills
Searchable directories	Filter by category, stars, update	awesome-skills.com, agentskill.sh

Official from Anthropic

Skill	What it does	Source
document-suite	Create Word, Excel, PowerPoint, and PDF with formatting and formulas	Anthropic (built in, paid)
theme-factory	Every file follows your house style automatically (colors, fonts)	anthropics/skills
brand-guidelines	Manage multiple brands and switch between them	anthropics/skills
skill-creator	Builds and optimizes your own Skills	anthropics/skills
mcp-builder	Generates a connector (MCP server) for you	anthropics/skills
webapp-testing	Tests your website automatically (Playwright)	anthropics/skills

Productivity and admin

Skill	What it does	Source
invoice / file-organizer	Sort and categorize receipts and documents	community (awesome lists)
web-asset-generator	Create favicons, app icons, and social images	community
markitdown	Convert PDF, PowerPoint, image, audio, and ZIP to text	community
seo-audit	SEO checklist (titles, meta, speed) as a to-do list	community
copywriting	Structured copy without the generic AI tone	community
obsidian-skills	Knowledge-base management (links, canvas) for Obsidian	community

Content, research, and web

Skill	What it does	Source
humanizer	Removes the generic AI tone, more human writing	community
content-research-writer	Writes with citations and built-in research	community
deep-research	In-depth research on a topic	community
agent-browser / browser-use	Automatically browse and gather information	community
epub-pdf-analyzer	Summarize and query ebooks and papers	community
notebooklm-skill	Turn videos and PDFs into a searchable knowledge base	community

For those who build

Skill	What it does	Source
superpowers	Turns Claude into a project manager (brainstorm, requirements, plan)	community favorite
get-shit-done (GSD)	Spec-driven work for large projects	TACHES
systematic-debugging	Debug like a senior: cause, hypotheses, fix	anthropics/skills

Skill	What it does	Source
git-worktrees / tdd-skill	Safe experiments and test-driven development	community
gstack (28 skills)	Claude as a virtual engineering team	Garry Tan
security skills	Code audits and finding vulnerabilities	Trail of Bits

For sales and customer contact

Skill	What it does	Source (credit)
contact / crm-enrichment	Enriches leads with web data (company, sector, signals)	SyncGTM, Use Apify
lead-researcher	Researches and qualifies prospects	Use Apify
email-drafter	Personalized cold emails and follow-up	Use Apify
crm-hygiene / crm-updater	Keeps your CRM clean and current	SyncGTM
pipeline-reporting	Pipeline report by stage	SyncGTM
proposal-builder / call-prep	Proposal and call prep from your CRM	community (8-skills-business)

For marketing

Skill	What it does	Source (credit)
seo-audit	Technical SEO analysis as a to-do list	Use Apify
content-pipeline	Research to draft to layout	Use Apify
social-media-manager	Posts per platform in your voice	Use Apify, Charlie Hills
email-campaigns	Welcome, retention, and win-back flows	George Hartley (Composio)
competitor-analyzer / brand-monitor	Tracks competitors and mentions	Use Apify
entrepreneur-skills (24)	Ads, copywriting, SEO, cold outreach	mfwarren

For admin and finance

Skill	What it does	Source (credit)
financial-reporter	Weekly or monthly financial summary	Use Apify
csv-data-summarizer	Automatic CSV analysis and patterns	coffeefuelbump
data-analysis-assistant	Statistics, visualization, hypothesis testing	liangdabiao
invoice / file-organizer	Sort receipts and invoices	community
compliance-checker	Checks text against your rules and policy	Use Apify

For HR

Skill	What it does	Source (credit)
Claude for HR (plugin)	Slash commands like /draft-offer and /comp-analysis in Cowork	AIHR
job-post-builder	Job postings with a fixed structure	community

Note: in a test of compensation benchmarking, a large share of the figures were well off. Use HR skills for drafts and structure, and always check hard numbers against a real source.

CRM, ERP, and SAP: connect instead of loose skills

For your business systems it is different: you usually connect Claude to them through a connector (chapter 14), not through a loose text file. What is ready to go in 2026:

System	How Claude connects to it
HubSpot	Ready-made connector, read and write, for all tiers with a paid Claude plan
Salesforce	Through Agentforce and MCP (Team or Enterprise)
NetSuite, Sage Intacct, Dynamics 365	Official MCP servers (including for finance)
SAP	Through MCP or AWS Bedrock (Joule), mainly for enterprise on SAP

For a small business the HubSpot connector is the most plug-and-play; SAP and Salesforce need more setup and a business plan. More about connectors and agents in your own systems is in chapter 14.

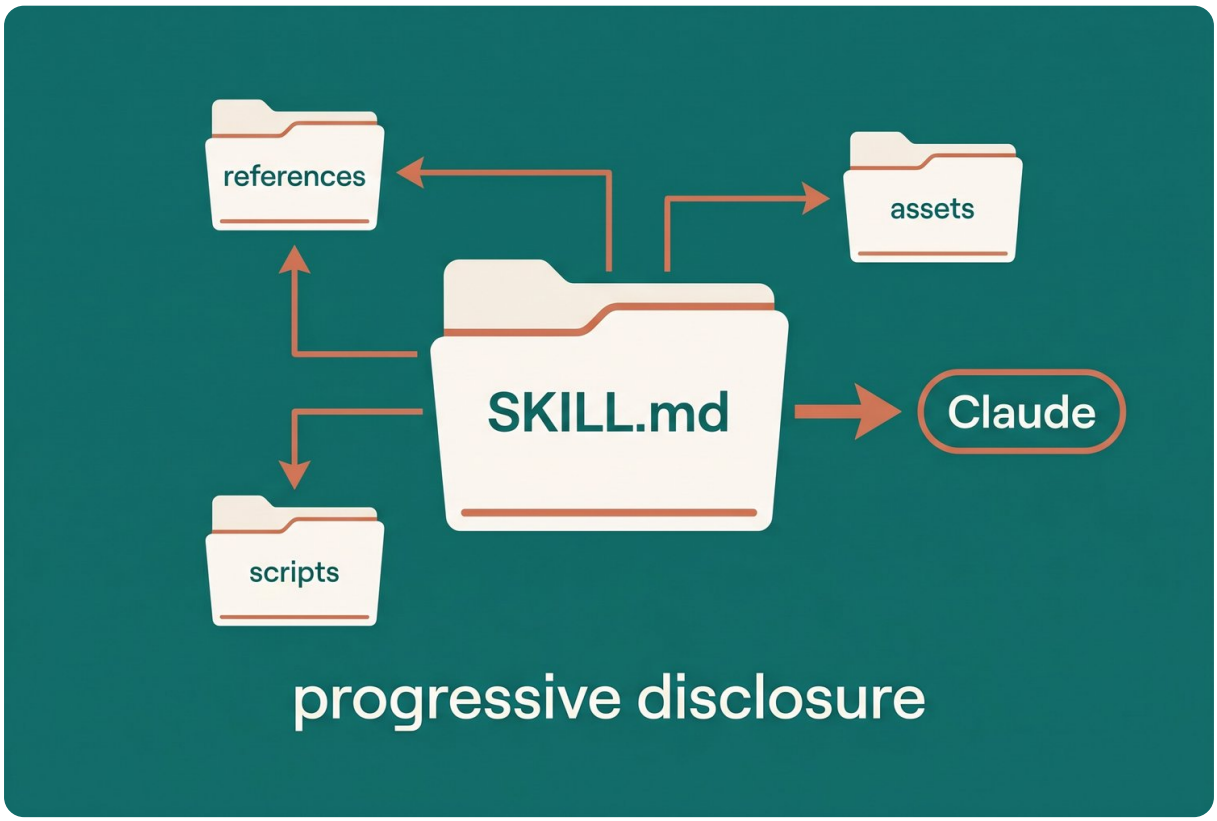
How to grab a Skill, in four steps

1. Open a collection (for example anthropics/skills or awesome-skills.com) and find your category.
2. Read the SKILL.md. It is plain text, not code that runs on your computer. Check the stars and the latest update.
3. Download the folder or use the install command from the repo (often an npx command or git clone into your skills folder).
4. Turn the Skill on in your settings and test it in a new conversation.

Safety note: community Skills are of varying quality. Read them before you use them, start with collections that have many stars and recent updates, and do not paste sensitive data until you know what a Skill does (chapter 7).

Do this now

Turn on Skills and the skill-creator in your settings. Pick the task you explained most this week, and have the skill-creator make a Skill of it: describe the task, answer the questions, and paste your best example. Test it in a new conversation. You then have your first piece of captured craft, and you understand at once why people consider this bigger than any other feature.



progressive disclosure. Clean, flat, one accent color.

In the next chapter we make the work run on its own: scheduled tasks and Routines, with which Claude carries out recurring jobs for you at a fixed time.

Scheduled tasks and Routines: work that runs on its own

A florist with four employees accidentally gained her best salesperson. She was losing customers after weddings and events: order once, then forget the shop. Someone set up an automated sequence in an hour that sends a personal reminder around birthdays and anniversaries. Three months later that was her best-performing channel. No team, no dashboard. One task that runs on its own.

That is what this chapter is about: the difference between using Claude, and having Claude work for you while you do something else. It is the fourth rung of the ladder, and for most small businesses this is where time comes back structurally.

What scheduled tasks are

You describe a recurring job once, and Claude carries it out automatically on the rhythm you choose. In the desktop environment Cowork that is called scheduled tasks, available on the paid plans since early 2026. Each task runs as its own session and may use everything you have set up: your connectors, your Skills, your files. The result is ready when you need it: a report, a briefing, a list of drafts.

The three flavors, briefly

- Cowork scheduled tasks. The path for non-technical users. You set them up in Cowork; they run locally while your computer is on and the app is open.
- Cloud routines. Run on Anthropic's infrastructure, even when your computer is off, and can even start on an event (a time, an incoming email).
- /loop in the terminal. For those working in Claude Code, for a short sequence within one session.

For a small business you start with Cowork scheduled tasks. Cloud routines are the step you take when you want something to run overnight or without your own computer.

How you set it up

Two ways. Type /schedule in a Cowork conversation, or click Scheduled in the sidebar and choose New task. You fill in: a name, the instruction (the prompt), the frequency (hourly, daily, weekly, on weekdays, or manual), optionally the model, and which folder Claude may work in. Done. From then on it runs without you thinking about it.

What a small business owner runs on a schedule

- A morning briefing that summarizes your email and calendar from the past day.
- A weekly report from your spreadsheets or shared folder.
- A monthly check on overdue invoices, with draft reminders ready.
- A competitor scan or news roundup in your field.
- Cleaning up and sorting a messy project folder.
- Customer touches around birthdays and anniversaries, like the florist.

It was already there when you woke up.

The golden rule: automate the boring, keep your hand on the important

The best automation, as someone put it nicely, is not an all-purpose agent that runs your whole business. It is a narrow system that pushes one tedious, repeated task forward, keeps track of what it did, and asks you for approval before anything important goes out. So automate the prep work: gathering, summarizing, preparing drafts. Leave the sending, deciding, and approving to a human. That way you gain time without losing control, and it fits exactly the transparency and care from chapters 7 and 8.

Pro tips from practice

- Describe the outcome, not the steps. "Deliver a list of overdue invoices with draft reminders every Monday" works better than a list of micro-instructions.
- Build the manual version until it is reliable first, then automate it. Automating a shaky process only makes it shaky faster.
- Use the fact that the output is kept between runs: a daily task can reference yesterday's, a weekly report builds on the week.
- Build in an approval step. A nice trick going around: have drafts appear in a Slack channel; thumbs up schedules it, thumbs down drops it. Emoji as a taste gate.
- Always set a review moment on anything that goes to a customer or touches money.

Common mistakes and how to avoid them

- Automating before it works manually. Fix: make it reliable first, then schedule.
- No human check on outgoing things. Fix: have drafts prepared, not sent.
- The stay-awake pitfall. A Cowork task runs only if your computer is on and the app is open; close the laptop and it does not run. Fix: adjust your energy settings, or use a cloud routine.
- Automating too much at once. Fix: start with one task, expand what works.

Ready to use: a morning briefing

Filled in here as an example; replace the details with yours:

```
Name: Morning briefing
Frequency: every weekday at 7:30
Instruction: Summarize what has come in since 5 PM yesterday in my email
and calendar. Give three things that need attention today, who I should
call or email back, and what I can let slide. Keep it to half a page.
Invent nothing; if something is unclear, say so.
Folder: my work folder
```

Ready to use: weekly invoice check

```
Name: Open invoices
Frequency: every Monday at 9:00
Instruction: Make an overview of invoices that are past due (from the
INVOICES folder or my accounting connector). Give the amount per customer,
the number of days overdue, and a friendly draft reminder message.
Send nothing; prepare the drafts so I can check and send them with one click.
```

Do this now

Pick one recurring job you do every week that does not need a judgment call each time, for example a briefing or a report. Set it up as a scheduled task with /schedule, have it prepare the output (not send it), and check for a week whether it is right. If it works, you have your first piece of work that runs on its own.

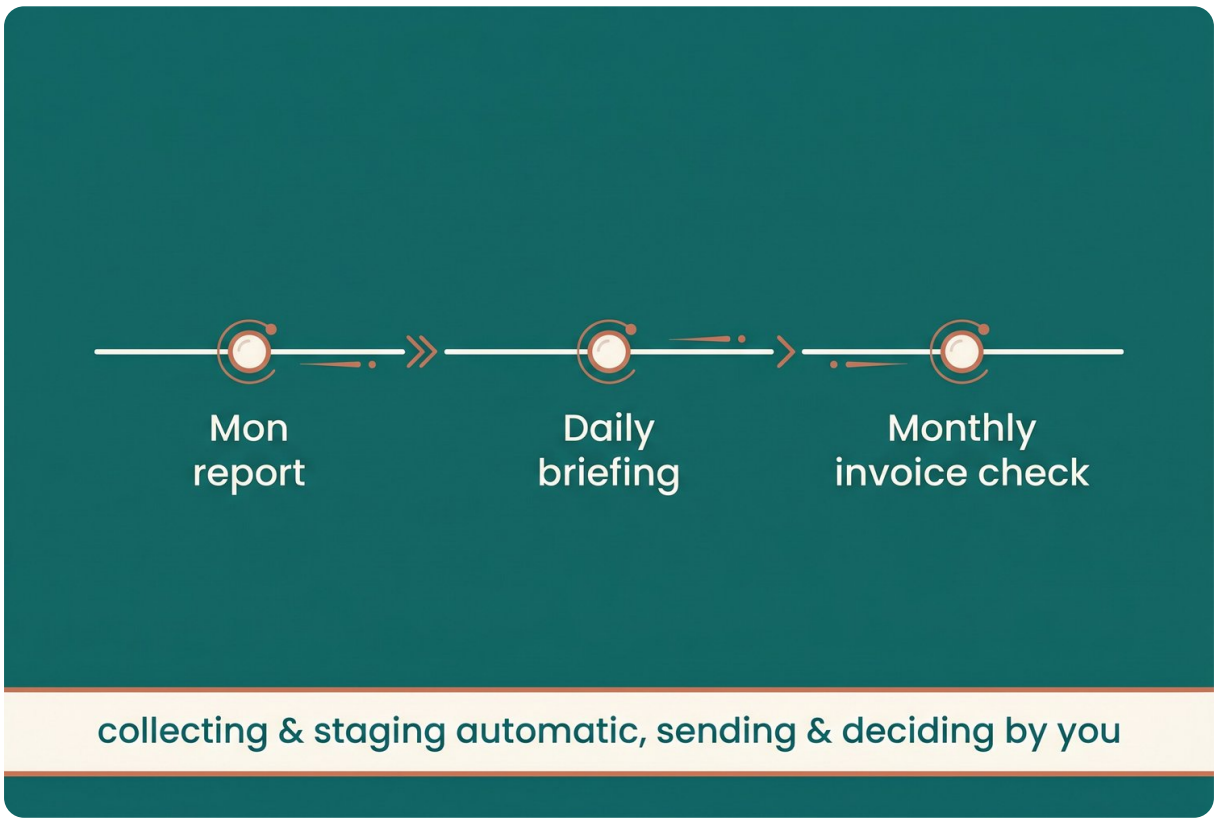


Figure - a week timeline with small tasks firing at fixed moments (Mon report, daily briefing, mont

In the next chapter we go to the top of the ladder: Claude Code, with which you turn an idea into a working tool, without a whole development team.

Claude Code: from idea to working tool

This is the top of the ladder, and the rung where Claude stops assisting and starts building. Claude Code is not a chat window but an AI agent that lives in a folder on your computer: it reads and writes your files, runs commands, carries out a job from start to finish, and can even set up its own tooling for you. You find it in the terminal, as a desktop app (the Code tab), and it shares a lot with Cowork.

You do not have to start here, and far from everyone reaches this rung. But once you know what it can do, you understand where this is headed: from "I ask a question" to "I have a tool built that does my work." This chapter is deliberately detailed, because this is where the real leverage is.

Not just for programmers

The biggest misconception is that Claude Code is only for code. The best summary from the community: Claude Code can build any tooling you need to make Claude Code work for your task, you only have to ask. People use it without writing a single line themselves for a knowledge base of hundreds of notes, processing meeting notes, tracking media, and even a complete podcast pipeline that does research, writes, edits, and adds voices. And yes, for software too.

The honest nuance: just as someone who knows nothing about building can lay a wall, but the result varies. Claude Code gives you enormous leverage, but it asks that you steer, plan, and check. That makes it powerful, not dangerous.

The building blocks (here is the power)

A few terms, because this is what makes it more than a smart chat:

- CLAUDE.md: your project's memory. A text file with who you are, what the project does, and your rules. Claude reads it at the start of every session. Keep it short; you can add a line by starting your message with a #, and reference another document with an @.
- Plan mode: think first, then do. In plan mode Claude may read, search, and investigate, but change nothing until you approve. The golden order is context, plan, code. You see the step plan and approve it (or adjust it) before anything happens.
- MCP connectors: connect Claude Code to your tools and data, for example your project system, a database, or your CRM. From then on it can pick up features from an issue tracker, query your database, or automate a workflow.
- Skills: your standard ways of working from chapter 11 work here too, as reusable expertise modules.
- Subagents: separate workers that do a subtask in their own context, for example combing through fifteen files, and return only the summary. That keeps your main session clean and sharp.
- Hooks: hard rules that always apply. For example: block a dangerous command, or only commit once the tests pass. These are the must-do rules alongside the should-do suggestions in CLAUDE.md.
- Plugins: bundles of skills, subagents, hooks, and connectors you install in one go.

From sketch to something that runs.

What you do with it

Here are the possibilities, with what people use them for in practice:

- Connect through MCP. Link Claude Code to your systems and have it really work in them: read data, create a task, pull a report. One connection, and you steer your software in plain language.
- Make your own tooling. Have Claude Code build skills, subagents, and small scripts for you. It sets itself up for your task; you describe what you want, it makes the configuration.
- Drive APIs and build integrations. Here it surpasses the standard connector tools. It builds integration logic with your company's rules baked in: "only sync if this is true, transform the data this way, and flag for review if this edge case occurs." That is exactly the kind of logic that off-the-shelf connectors break on.
- Pull in your own documentation. Have it fetch the current docs of a tool or your internal manuals, so it works on what is true now instead of old knowledge. There are skills that do exactly that.
- Bulk import and export. Read in a whole folder or dataset, process everything in one run, and export in bulk. The underlying technology is built for this kind of batch work, so hundreds of records at once is no problem.
- From idea to product. A proof of concept in an afternoon, and with guidance even working software, without a whole development team.

How to start, even without a technical background

You do not need years of experience, just the willingness to steer. The shortest path:

- Install Claude Code, or use the Code tab in the desktop app.
- Make a folder for your project and open Claude Code in it. That folder is your workspace; all settings live there.
- Have it set itself up. Ask: "Make a CLAUDE.md for this project, ask me questions about my way of working first." Then ask for a fitting output style, a skill or two, and optionally a subagent.
- Work in plan mode for anything non-trivial. Have it show a plan first, approve it, and only then have it execute.

The nice part: Claude Code can talk you through all of this. If you do not know something, ask Claude itself.

Honest and safe

- Plan first, then execute. Do not start with "build me an app"; that gives chaos. Start small and concrete, with a plan you approve.
- Have it check its own work. Give it a test it can run (a test, a check, a comparison), so it stops when the work is right and keeps going until it passes.
- Keep your hand on the controls. Dangerous commands and reading passwords should be blocked; safe, frequently used commands you approve once. That way you build trust without risk.
- Sensitive data stays within your own environment (chapter 7), and a human approves what goes out.

The non-obvious examples

- An internal helper that reads your open invoices from a CSV, drafts a reminder per customer, and prepares everything for your approval.
- An integration that drives your CRM or an ERP environment through a connector: read and update data under your permissions (chapter 14).
- A documentation pipeline: pull the structure or docs of a system, and import or export hundreds of records in bulk.
- A company brain that grows with you, plus a second model (such as Codex) that reviews Claude's work, so two pairs of eyes see every change.
- For those with software: Claude Code as a step in your release pipeline (a GitHub action), with strong guardrails and review.

Pro tips from practice

- Turn on plan mode for anything more than a trivial change. It saves "that is not what I meant" moments.
- Keep CLAUDE.md short (under two hundred lines) and reference detail files instead of cramming everything in.
- Always give it a way to test its own work.
- Use subagents for heavy digging, so your main session stays sharp.
- Have it configure itself: "make a skill for this" or "put this as a standing rule in CLAUDE.md."
- Work with versions or backups before you have big changes made.
- Load connectors lazily. By default every MCP connector loads its tools in every conversation, even if you do not use it, and that costs room. Turn on lazy loading so a connector only loads when needed (a tip from the Claude Code community).
- Sometimes a Skill beats a connector. For recurring work a Skill (chapter 11) often costs less room than an MCP connector that always runs along; a common rule of thumb among heavy users is "Skill over connector where you can."
- Do not give a blank check. The maker of Claude Code prefers to approve a list of safe commands once and capture that in his settings (shared with the team) rather than turn off all the checks (Boris Cherny). His most important tip: always give Claude a way to test its own work, which doubles to triples the quality.
- A test prompt that works well in practice: "double-check every claim you just made, and at the end put in a table what you could and could not verify."

Common mistakes and how to avoid them

- "Build me an app" with no plan. Fix: plan mode, start small, approve.
- No check step. Fix: give it a test or check, and review the work.
- Cramming everything into one endless session. Fix: subagents for subtasks, a clean main session.
- No backup for big changes. Fix: version control or a copy beforehand.
- Leaving dangerous commands unguarded. Fix: block them with a rule and only approve safe ones.

Ready to use: a CLAUDE.md for a business folder

Filled in here as an example; replace the details with yours:

```
# CLAUDE.md - Van Dijk Climate automation

Goal: this project automates my quote and invoicing work.
Folders: QUOTES/, INVOICES/, CUSTOMERS/, OUTPUT/

How to work:
- Plan first, then execute. Ask approval for anything that gets sent or
  deleted.
- Write new output to OUTPUT/; do not touch the source folders.
- Follow the tone and rules in references/style.md.

Rules:
- Never send customer data outside; work only within these folders.
- Do not commit or send anything automatically; show changes first.
```

Ready to use: a build instruction in plan mode

```
I want a small internal helper that [FILL IN: for example reads my open
invoices from a CSV and drafts a friendly reminder per customer].
Work in plan mode: investigate first, ask me the questions you need,
and show a step plan before you build anything. Only begin after my approval,
and prepare the output for review instead of sending anything.
```

What people actually build with it

To show the breadth, a cross-section of what people make with Claude Code, from down-to-earth to surprising.

For the business:

- A complete quote and calculation system for custom work, linked to materials, the workshop, and suppliers.
- Internal company apps that replace expensive licenses.
- Marketing work: brainstorming angles, rewriting landing pages, and tracking project status through skills.
- Automations for email, invoicing, and follow-up.

Personal and creative:

- A podcast pipeline that does research, writes, edits, and adds voices, with local tools where possible and connectors where needed.
- Custom little apps for a specific problem: a home poker tournament manager (built the night before the tournament), a groceries app with a shared list, a health app.
- A multi-layer system to organize your work and life.

Surprising:

- Deciphering two-hundred-year-old French handwriting for genealogy research.
- Generating CAD drawings.
- Legal and financial prep work: preparing a tax return, appealing an assessment, or organizing a file like a forensic accountant.
- Lesson plans and teaching material for teachers.

The thread: these are rarely ready-made products in one click. They are people who had an idea, started small, and steered step by step. And the honest flip side: anyone who shouts "just build this" with no plan gets a messy heap back, with duplicate features and code in odd places that you can no longer take over yourself. That is why the lesson of this chapter stands: plan first, cut into small pieces, and keep your hand on the controls. The difference between a fun experiment and something that really runs is mostly in how well you steer.

One last practical tip that comes up everywhere: a connector hub like Rube links Claude to hundreds of apps at once (Slack, GitHub, Notion, and more), so you do not have to set up a separate connector for each service. More about connectors in the next chapter.

MCP: how Claude gets eyes and hands

Until now Claude was smart but blind. It could think about what you paste in, but it could not look into your systems or change anything itself. MCP is what flips that. The sharpest summary from the community: without MCP Claude is smart but blind, it can analyze code you show it, but it cannot see your systems and do anything in them; with MCP Claude gets eyes and hands (after a widely shared explanation by developer Chayan Bhattacharya on dev.to).

MCP stands for Model Context Protocol, an open standard Anthropic released in late 2024. Think of a USB-C port for AI: where every device used to have its own plug, now there is a universal one. MCP is that universal plug between Claude and the outside world. Instead of building a separate, brittle connection for every tool, everything talks through the same protocol.

Under the hood it works like this. An MCP server is a small piece in between that offers two things:

- Tools: concrete actions Claude may perform. Think "create a task," "read this database," "open a repair request." Claude calls a clean, structured instruction and gets a clean answer back.
- Resources: data Claude may read. Your documents, your records, your project history.

The important difference with "Claude reads a website for me": MCP does not scrape screens and does not guess. It talks directly to the source through an agreed language, gets reliable data back, and reasons about it just like about anything you type in. That is why it is so much more stable than a bot clicking through a web page.

A few things that make it immediately practical for a small business:

- One connection, plain language. Connect your project system, your database, or your CRM once, and after that you steer it with sentences: "queue these five tasks," "pull last month's revenue," "update this customer record."
- Ready-made servers already exist. Anthropic and the community provided servers for Google Drive, Slack, GitHub, Git, and databases from the start. You do not have to reinvent the wheel.
- Open standard. The same MCP server works not only in Claude Code, but also in other AI tools. You invest in a connection, not a dead end.
- Local or remote. An MCP server can run on your own machine (for sensitive, internal data) or in the cloud. You decide where your data passes through.
- Always under your permissions. A connection can never do more than you may. You approve access, and you can put write actions behind an approval.

This is the pivot point of the whole ladder. An ordinary conversation only becomes an employee when it can look into and act in your real systems. MCP is the technology that makes that possible, and in the next chapter we set one up step by step.

GitHub: give your work a time machine (and get it off your laptop)

There is one connector I single out for everyone who builds with Claude Code, even if you never become a programmer: GitHub. Not because it is hip, but because it solves two problems you will definitely run into the moment an AI agent starts working in your files.

First the terms, in plain language. Version control is a timeline of your project. Every time you lock in a point (a "commit"), you make a snapshot you can always return to. Git is the engine that does that locally on your computer. GitHub is the place in the cloud where that history also lives. The comparison that makes it click: just as OneDrive syncs your Word documents across multiple devices, GitHub syncs your project history and puts everything in a central place (from the teaching materials of NBIS, a Swedish bioinformatics institute).

Why this matters, especially with an AI agent that changes things on its own:

- A real undo button. Claude changes ten files and one goes the wrong way. With version control you roll back to the previous snapshot in seconds. Without it you lose your work. This is your safety net.
- Not everything on your laptop alone. A stolen or crashed laptop without a backup means: gone. With GitHub a complete copy sits safely in the cloud, including the entire history.
- Working from any computer. Start at the office, continue at home. You pull the latest version and everything is there, exactly as you left it.
- Safe experimenting with branches. A branch is a separate line where you dare to try something without touching your working version. If you like it, you merge it. If you do not, you throw the branch away and nothing happened.
- Who changed what, and why. Every snapshot has a short note. You see the whole history: what changed, when, and for what reason. If something is broken, you trace when it crept in.
- One source of truth. No more "quote_final_v3_reallyfinal.docx." One central, current version that you and any colleagues work around.
- Collaborating without overwriting each other. Someone else proposes a change through a pull request: a proposal you review and approve first before it enters the main version.

In short: version control gives you a complete history and the certainty that you can always go back, and that is exactly the peace of mind you want when you set an AI loose on your files. It is not a luxury for programmers; it is the insurance that makes "just build this" safe.

The GitHub MCP: let Claude work in your repositories

Once you use GitHub, the GitHub MCP server comes into play, GitHub's own official connector. It gives Claude eyes and hands in your repositories: it can read and update your projects, changes, tasks (issues), and proposals (pull requests), with dozens of actions (the official github/github-mcp-server has more than forty). In practice that means sentences like:

- "Summarize what changed in this project this week."

- "Create a task for the bug I just described."
- "Open a proposal with this change, so I can review it before it goes in."
- "Review this proposal and point out risks."

A practical habit of heavy users: allow "pull" (fetching) automatically, but put "push" (writing to the cloud) behind your approval, because that is riskier. And if something breaks, have Claude use a built-in GitHub trick (git bisect) to pinpoint the exact change that broke it. That way version control becomes not something you keep up on the side, but something Claude manages for you while you steer in plain language. A second connector you often hear: a connector hub like Rube links Claude to hundreds of apps at once (Slack, GitHub, Notion, and more), so you do not set up something separately for each service.

Below is a filled-in instruction you can paste. The situation is already filled in for you; replace the italic parts with yours.

```
I want to bring my project folder under version control and connect it to
GitHub, so I have a safe history and not everything sits only locally.
My project: a folder with quotes and invoices for my HVAC business
[FILL IN: short description of your project].

Do this in plan mode:
1. Explain to me in two sentences what a commit and a branch are.
2. Set up version control in this folder and make a first snapshot with a
   clean note.
3. Explain how I connect this to GitHub and what is then safely in the cloud.
4. Propose how we set up the GitHub connection (MCP), so you can later
   create tasks and proposals for me.

Do not change anything in my files yet. Show the plan first and wait for
my approval.
```

This is exactly what the next chapter builds on.

Do this now

Pick one small, concrete helper that would save you work, for example something that reads in a list and turns it into clean text. Open Claude Code in a new folder, have it draft a CLAUDE.md for you, and give the build instruction in plan mode. Approve the plan, have it build, and review the result. You do not have to know how to program; you only have to steer.

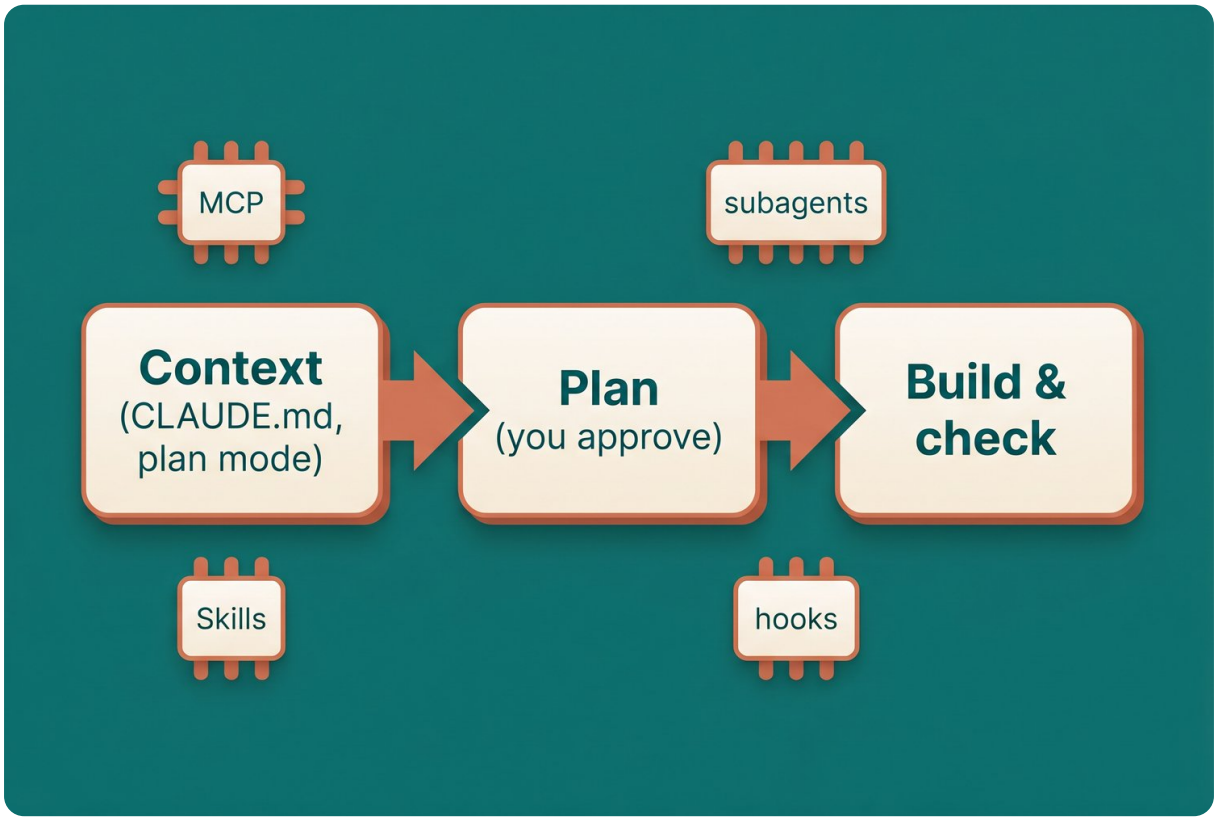


Figure - the workflow in three steps: context (CLAUDE

In the last chapter of this part we close the ladder: connectors and managed agents, with which an AI agent works in your own software (your CRM, your accounting, your ERP environment) under your supervision.

Connectors and managed agents: an agent in your own systems

This is the highest rung of the ladder. In chapter 13 Claude learned to build; here it learns to run alongside. We go from "Claude can make documents" to "Claude works in your CRM, your accounting, your project system, and your ERP environment, under your supervision." The technology underneath you already know from chapter 13: MCP, the universal plug. This chapter shows how you flip that plug once and what becomes possible, plus the rules you may not skip the moment an AI is allowed to look into and act in your real systems.

Two tracks come together here. As a user you want to know which connections make your work lighter and how to turn them on. As an owner you want to know what is safe, what it costs, and where the limits are. Both are covered, and both start in the same place: a connector.

Connectors: the plugs you flip once

A connector is a ready-made connection between Claude and a service you already use. Once connected, Claude recognizes on its own when a linked tool is relevant. Ask about your calendar and it pulls in your calendar without you having to name the connector. You steer in plain language, the connection does the rest.

Let me clear up a misconception right away: this is not a programmer's tool. The most common user of such connections is not a developer hooking up a design tool, but someone who says "I want to get at my own company's numbers" and pulls reports out of it (an observation from an in-depth interview in the Pragmatic Engineer newsletter, 2026). That is exactly the small-business user. And it is no longer niche: there are now thousands of connections available and the standard is used tens of millions of times a month, with the majority of users planning to do more with it (figures via Goodcall, early 2026). The same connection also works just as well with other AI assistants, so you are not betting on a dead end.

There are two flavors:

- Directory connectors. Pre-vetted connections from Anthropic's official list (now more than 75). One click and you are connected. Think Google Drive, Gmail, Calendar, Slack, Notion, and more.
- Custom connectors. Your own connection to any service that has a so-called remote MCP server, including internal tools not on the list. Available on Free, Pro, Max, Team, and Enterprise (on Free a maximum of one), and at the time of writing in beta.

You find them through Customize, then Connectors. Or click the tools icon under the input field and choose Browse connectors. Connecting is a plus button and one login.

Two things to know up front:

- The email connectors are read-only by default. Claude reads your email to pull context; it does not send on its own (Gmail can now prepare drafts). That is exactly the safe default you want.
- The real work starts with stacking. One connector is handy, several together is where the gain is. "Check my email for meeting requests, check my calendar for availability this week, and prepare draft replies with suggested times." That is your email and calendar connectors working together in one instruction.

A down-to-earth approach I keep seeing in practice: do not connect everything at once. Start with the two or three tools you touch daily (for most people: email, calendar, and your documents folder), build a few working routines around them, and expand after that (advice from a practical connector guide by Jamout, 2026).

Connect once, then steer with language.

Adding your own connection

If you have a service that is not on the list, you add it yourself. The steps, in plain language:

- Go to Customize, then Connectors.
- Click the plus button next to Connectors and choose "Add custom connector."
- Fill in the name and the URL of the remote MCP server.
- Optional: under the advanced settings, fill in an OAuth Client ID and secret if the service asks for it.
- Click "Add" and then go through the same login and approval step as for a regular connector.

An important detail that surprises many people: a custom connector connects from Anthropic's cloud, not from your own computer. Even if you use Cowork or the desktop app. So the server has to be reachable over the public internet. If it sits behind a company firewall or on a closed network, you have to allowlist Anthropic's IP ranges, so a secure, outbound-only connection is created. Local MCP servers (through the desktop app's config file) are a separate mechanism and do not work in Cowork or on the web.

If you work in a Team or Enterprise environment, only an Owner adds a custom connector through the organization settings. After that colleagues see it in their list and only have to click "Connect." That way you keep control as a company over what gets connected.

The GitHub connection, now actually set up

In chapter 13 I explained why GitHub is your safety net: a time machine for your work, a backup off your laptop, and the place where you can safely go back when an AI agent changes something. Here we set up the connection.

GitHub has provided an official remote GitHub MCP server since September 2025, generally available. It works with a modern, secure login (OAuth, with short-lived tokens that refresh themselves, safer than a fixed key set by hand) and works with any MCP host: Claude, Claude Code, Cursor, and more. You do not have to install anything; you add it as a custom connector with the server's URL (the official remote server runs at GitHub's address for MCP), and you go through the login in your browser.

What Claude can then do for you, in plain sentences:

- "Summarize what changed in this project this week, and who did what."
- "Create a task (issue) for the bug I just described, with a clear title."
- "Open a proposal (pull request) with this change, so I can review it before it goes in."
- "Review this proposal and point out risks or missing checks."

The nice part: that way version control becomes not a separate chore, but something Claude keeps up for you while you steer in plain language. The more than forty actions of the GitHub server cover nearly all of your daily GitHub work: projects, changes, tasks, proposals, and releases (source: github.blog changelog, remote GitHub MCP Server GA). The main benefit GitHub itself names is the end of switching back and forth between tabs: your agent works directly with your GitHub data through clean, structured actions.

Which connections pay off most

You really do not have to connect everything. Below is a focused map for a small business, with where each connection excels. Start with the top row; expand as a routine proves itself.

Connection	For what	Typical instruction
Email, Calendar, Drive	Your daily base	"Schedule these three appointments and email the confirmations."
Slack or Teams	Internal communication	"Summarize today's #client channel discussion."
Notion or your knowledge base	Knowledge and notes	"Update the project page with the decisions from this conversation."
Monday.com or your project system	Tasks and pipeline	"Create tasks for the action items and assign them."
CRM (HubSpot, Salesforce)	Customer data, leads	"Show this month's marketing pipeline by campaign."
Accounting or invoicing	Admin	"Pull the open invoices and draft reminders."
GitHub	Version control, projects	"Open a proposal with this change for my approval."
Make or Zapier (hub)	Thousands of apps	"Start the quote flow if this condition is met."

Two practical routes if you want to go broader without setting up something separate for each service. The first is a connector hub like Rube (from Composio), which links Claude to hundreds of apps at once. The second is the automation connectors Make and Zapier, which themselves unlock thousands of apps; handy when the service you want has no connector of its own (overview via SegmentStream and Cal.com, 2026). Note: many CRM connectors are still read-only or in beta, so check per connection what is allowed before you build on it. One reason this is more durable than the old screen robots that mimic a button: a connection works under the screen layer, at the level of the data itself. Move a button a few pixels and a screen robot breaks; a connection keeps working (from a connector overview by Rainstream, 2026).

Managed agents: from a single task to an employee that keeps running

Until now Claude did a task while you watched. An agent is the next step: it takes an instruction, works through a series of steps, uses the tools you gave it along the way, and comes back with a result. In chapter 12 you saw the simple variant (scheduled tasks and routines). Here it is about the heavy variant.

Anthropic launched Claude Managed Agents on April 8, 2026: a hosted service that runs the entire engine room of an agent for you. Sandboxing (a walled-off workspace), preserving state between steps, executing tools, granting limited permissions, and observability are all included. The core, in Anthropic's own words: decoupling the brain from the hands, so you can build an agent without assembling all the infrastructure yourself. Early users include Rakuten, Notion, and Sentry.

For the full picture, two related terms:

- The Agent SDK is the library variant: you run the agent in your own environment and keep everything local. Out of the box you get file editing, command execution, web search, a reason-and-act loop with human approval moments, subagents, sessions, rollback points, and MCP support. Good when you want full control or run inside an existing pipeline.
- Managed Agents is the hosted variant: Anthropic runs the agent and the sandbox, your application sends instructions and gets results back. Good when you want to go from prototype to production fast without managing infrastructure yourself.

What makes this so interesting for those who want to keep data within their own walls: since May 2026 you can use self-hosted sandboxes and MCP tunnels. With those, the agent's execution runs on your own infrastructure, while the steering stays with Anthropic. Your data stays within your own perimeter; only the direction runs outside. For a small business with sensitive customer or financial data, that is exactly the switch that makes the difference between "interesting" and "responsible."

Be honest about where this stands. Managed Agents is developer tooling and runs in public beta, priced at the normal token costs plus an amount per session-hour. Some of the smartest features (such as agents that first practice against historical data, get a measurable success condition, or collaborate in a team on a workflow) are still research preview. As an owner you do not have to build this yourself; the point is that you know it exists, so you can have it built or buy it specifically when a process is worth it. From June 15, 2026, use of the Agent SDK on subscriptions also draws from a separate monthly allotment; good to know for anyone who really starts deploying it.

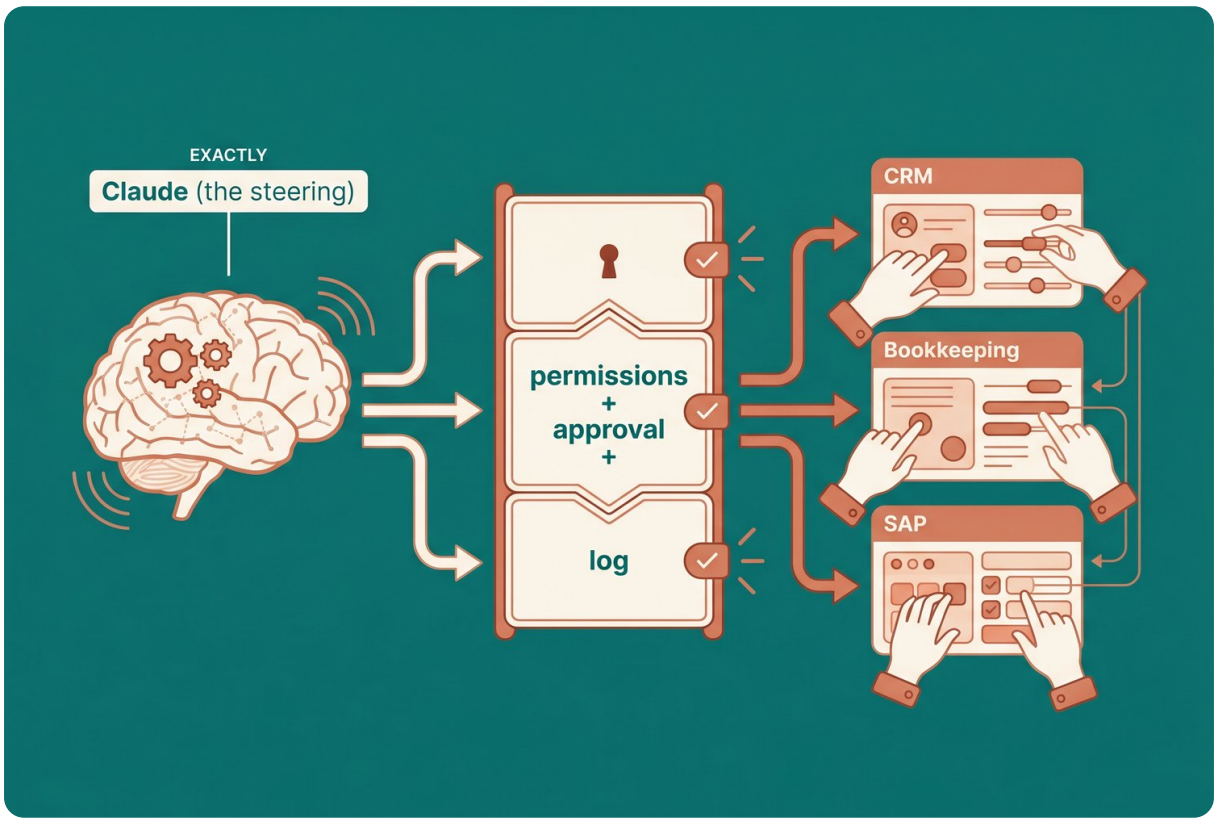


Figure - "the brain and the hands"

Claude, ERP, and your CRM

This is doable, for anyone who works with it. At SAP Sapphire 2026, SAP and Anthropic announced an expansion of their partnership: Claude becomes a central reasoning and agent layer within the SAP Business AI Platform, extending the capabilities of Joule (SAP's AI assistant) and the Joule agents (source: SAP News Center, May 2026). The connecting language underneath is, again, MCP. With it, agents pull context and act across SAP systems such as S/4HANA, SuccessFactors, and Ariba. SAP HANA Cloud gained full MCP support in early 2026, so Joule agents can reach the database directly (source: ERP Today).

The tone-setting quote is from Daniela Amodei (Anthropic): Claude is built for the work that keeps a business running (closing the books, rerouting a delayed order, approving an expense), and with Claude on the SAP platform that work happens inside the systems businesses have already invested in, with the trust and governance SAP customers are used to. SAP CEO Christian Klein calls it the foundation for the "autonomous enterprise": AI that knows the business context and operates within the existing control mechanisms.

A few concrete things that come out of this, including the lesser-known:

- The SAP AI Agent Hub is now in every SAP Business AI Platform subscription with Joule Studio. There you discover agents across your whole landscape and manage up to 500 SAP and non-SAP agents, with the guardrails to govern them (source: SAPinsider). There are already more than thirty specialized agents and over 2,500 Joule Skills, which collaborate across SAP and non-SAP through an agent-to-agent protocol (SAP News, Q1 2026).
- The new Joule Studio is SAP-managed: no configuration, runs out of the box, with audit logging and data protection built in by default. Agents' memory is kept in HANA Cloud, so context persists between sessions. You plug MCP servers into it to bring in SAP context, and for those who want to work technically it works together with tools like Claude Code and Cursor, with everything landing in Git and your CI/CD pipeline (source: The New Stack, May 2026). That last point ties in seamlessly with the version control from chapter 13.

A concrete picture of what this yields: a treasury manager asks Joule to prepare a presentation for the CFO ahead of a meeting with a bank. Within minutes there is a finished presentation with current figures, analysis, and points of attention about financial risks, work that previously took hours by hand (example from ERP Today).

The work that keeps a business running, inside the systems you already trust.

The same pattern holds for your CRM, even if you do not run SAP. Major vendors now publish official MCP servers, so Claude can read leads, tasks, and customer history (and, where allowed, update them). The real value is not in one system, but in combining: pipeline from your CRM, enrichment from a data source, call notes from your knowledge base, all in one Claude conversation that reasons about it. As a much-cited explanation puts it: the bottleneck is not the technology, but the governance (MCP for Enterprise, Botscrew 2026).

One nuance I name deliberately because it counts commercially: in the SAP Integration Suite all APIs and flows can be converted into MCP tools, but "can be converted" is not the same as "is production-supported today." That distinction decides whether you dare to build a business process on it (nuance from SAPinsider). And that is exactly where the questions sit that an experienced advisor asks and a demo does not: who maintains those connections, and what happens when one breaks? If your purchasing agent leans on live connections between S/4HANA, Ariba, SuccessFactors, and a supplier portal, a broken connection does not just slow the process, it can halt it. So agree up front with your vendor or implementation partner about who owns the testing, monitoring, and recovery (nuance from ERP Today and SAPinsider). And keep in mind: deeper integration with one model also means deeper dependence on that vendor's roadmap and pricing. No reason not to do it, but a reason to do it with eyes open.

Finally, an honest note from practice. A developer who set Claude loose on an ERP environment through MCP noticed that it sometimes made changes directly in the system without touching the local file (from a candid log in the developer community). That is

exactly why the rule of this chapter applies doubly in an ERP environment: start by reading, grant write permissions sparingly, and put a human on the button for anything that touches a transaction. How to set that up safely for every connection is below.

Connecting safely: the non-negotiable rules

The moment an AI is allowed to look into and act in your real systems, the playing field changes. Not to scare you, but to keep you sharp: in the first months of 2026, dozens of vulnerabilities in MCP servers were reported, and thousands of poorly secured servers sit open on the internet. The technology is young. That does not mean "do not do it," it means "do it like a grown-up." Five rules you do not skip:

- Least privilege. Give a connection only what it needs. Start read-only. An agent that answers customer questions may read in your CRM, but not delete. An agent that views invoices may read, but not start payments.
- A human approves irreversible actions. Deleting, transferring money, changing in bulk, sending something out: those are moments where a wrong or misled instruction does real damage. Put a hard approval step in front of those. This is the same human-in-the-loop that runs through this whole handbook.
- Have the agent act under your permissions, not admin permissions. The safe approach is that an action uses exactly the rights of the person who started it, not an all-powerful key sitting on the server. If someone cannot do something elsewhere, the agent cannot do it on their behalf either.
- Do not trust a connection blindly. Connect only servers from parties you trust; custom connectors are not verified by Anthropic. Be alert to a subtle but real risk: a malicious or infected source can smuggle in hidden instructions (prompt injection). Treat every external connection as untrusted in principle until you know otherwise.
- Keep a log. Who did what, when, and through which connection. Without visibility you do not know if something went wrong. With a log you can look back and steer.

Two things even experienced users forget, and that make the difference (from a security guide for Cowork by Harmonic Security, 2026):

- Check your write connections separately. Connections that send or create something (send mail, post a message, create a file) are the riskiest. Turn them off unless you really need them, and keep them on a separate list.
- Treat a shared "live" page like a macro file. An interactive page (an artifact) you get from someone runs in your session and calls connections with your permissions, without you seeing which. Open such a shared page from an unknown source as carefully as an Excel with macros.

Tie this to chapter 7: sensitive data belongs in an environment that does not train and that respects your perimeter (the business layer, not the consumer layer). The self-hosted sandboxes and tunnels from the previous section are the technical implementation of this. The rule of thumb stays simple: start small, start reading, and only open up more once you trust it.

Pro tips from practice

- Connect at most two or three tools to start, and build working routines around them before you expand.
- Ask a new connection first what it can do: "Which actions do you have available through this connection?" Then you know the limits right away.
- Put write actions (create, change, send) behind your approval by default; leave reading free.
- Use a hub (Rube, Make, or Zapier) only when you need several separate services; for one or two tools a direct connector is clearer.
- Keep a short list of approved connections for your business, just like your AI inventory from chapter 8.
- If you really want to keep data within your walls, look at the self-hosted/tunnel variant instead of an ordinary cloud connection.
- Clean up monthly: turn off connections and Skills you do not use. Too many active connections can confuse Claude (it sometimes loads one you did not mean) and costs unnecessary room in your limit (a tip from the Cowork community).

Common mistakes and how to avoid them

- Connecting everything at once. Fix: start with your daily two or three, expand on evidence.
- Granting write permissions right away. Fix: start read-only, add writing after an approved path.
- Putting an all-powerful key on the server. Fix: have actions run under the user's permissions, not admin.
- Trusting an unknown custom connector. Fix: connect only servers from parties you know; treat the rest as untrusted.
- No approval for irreversible actions. Fix: hard human-in-the-loop for deleting, paying, sending, and bulk changes.
- Assuming "can be connected" is the same as "production-ready." Fix: validate the read path first, build write actions only after.
- Leaving dozens of connections on "just in case." Fix: monthly cleanup; only on what you use.

Ready to use: have a safe connection plan drawn up

Below is a filled-in instruction you can paste. The situation is already filled in for you; replace the italic parts with yours.

```
I want to connect Claude safely to my business systems and want a plan first,
not loose connections. My situation: an HVAC company with 8 employees
[FILL IN: your business and size]. I use Gmail, Google Calendar,
Google Drive, and Monday.com daily, plus an accounting package and a CRM
[FILL IN: your tools]. Sensitive data: customer data and invoices
[FILL IN: what is sensitive for you].

Draw up a phased connection plan:
1. Which 2 to 3 connections do I set up first, and why those?
2. Which connections do I keep read-only, and which actions must really be
   behind my approval (create, change, send, pay)?
3. Which sensitive data may not go to an ordinary cloud connection, and where
   do I choose a walled-off variant?
4. How do I keep visibility: what do I log and put in my AI inventory?
5. Give me one concrete first routine per connection that saves time.

Give the plan as a short table with phases. Ask me the questions you need
before making assumptions.
```

Do this now

Open Customize, then Connectors, and connect exactly one tool you already use today, for example your calendar or your documents folder. Then give Claude one instruction that uses that connection and that really saves you something ("check my calendar and suggest three slots for a meeting this week"). If that feels good, connect a second one and have them work together. Build out from there, reading first, and put write actions behind your approval.

With that, Part 4 closes. You now know the whole ladder: from a first conversation, through documents, projects, Skills, and scheduled tasks, to an agent that works in your own systems under your supervision. In Part 5 we make it concrete per role: what this means for the owner, for sales, marketing, operations, service, and for visuals and presentations. There we translate all the building blocks from this part into the work you do every day.

For the owner and leadership

So far this handbook has been about what Claude can do and how to use it safely. This part makes it concrete per role, and we start with yours: the owner or executive who decides, sets up, and safeguards. Your job is not "to do something with AI." Your job is to make sure AI delivers measurable value without unnecessary risk, and that your people actually use it.

Let us start with an uncomfortable number, because it is actually good news for you.

The number nobody says out loud

A much-discussed MIT study from 2025 (The GenAI Divide: State of AI in Business, led by Aditya Challapally of the MIT Media Lab) reaches a hard figure: roughly 95 percent of generative-AI pilots in companies produce no measurable result on the bottom line. Companies have poured tens of billions into it, and only about five percent creates real value.

Before you tune out: the reason why is the good news. According to that same study it is not the technology, the budget, or the talent. It is a learning gap. Most failures are tools that look good in a demo but do not integrate with your work and your data, do not learn from corrections, and do not connect to a real process. In other words: they are organization and approach problems, not model problems. And those are exactly the things you, as the owner, can influence.

A fitting image from a discussion about this: deploying an AI with no context is like sending a toddler to the fridge for a drink. Sometimes you get a soda, sometimes water, sometimes a jar of mayonnaise. Online that is funny; in your books it is fatal. The whole rest of this chapter is about how you end up in that five percent instead of the ninety-five.

Do not start with the tool, start with the process

The biggest mistake is buying a tool and then going looking for a problem. In early 2026 there are more than ten thousand AI tools in hundreds of categories; the marketing makes it nearly impossible to tell value from hype. So flip it around: start with one process that hurts.

A simple way to choose: score your processes on two axes.

- Impact: how much time, money, or missed revenue is in it? How often does it happen?
- Feasibility: is it repeatable and pattern-based? Is the data in order? Are the rules clear?

Take the process with the highest combined score. For most small businesses that sits in the back of the shop (quotes, invoices, email, scheduling, retyping data) or in first-line customer contact. Precisely there, at the spot with the most friction, the gain is most direct (a finding that recurs in the MIT study and in a Forbes analysis of it: successful companies focus on the less glamorous back-office work).

An experienced voice from practice (a B2B owner in a Reddit discussion) sums up the gain plainly: it is rarely the spectacular things. It is searches that come up empty less often, data from emailed PDFs and spreadsheets that lands in your system automatically, and first-line customer questions that get handled on their own. Boring, and valuable precisely because of it.

Not everything at once. First the process that hurts most.

What does it really return

The boardroom question has shifted from "should we invest in AI?" to "prove it returns something." The honest answer: AI often returns a lot, but you have to measure it, otherwise the value disappears into the gap between purchase and use. The adoption numbers are striking: in the Netherlands, SME use of AI roughly tripled in a year, from about 6 to 33 percent (Exact SME Barometer), and Statistics Netherlands sees firms with ten or more staff climbing from around 14 to 23 percent. Dutch SMEs also lead the continent in intent: 84 percent plan to invest more in AI over the next three years. Yet 94 percent of owners feel their own business is still not automated enough. The gain is not in buying, it is in really using it.

So measure on three layers, in plain language:

- Time and effort. How much faster is a task? How many hours a week does a team save? (One in five SMEs that use AI already save around twelve hours a week; for marketing work alone, some report 5 to 15 hours.) How much less retyping and rework?
- Quality. Is the outcome as good or better than before, measured against a baseline? How often does a human still have to step in?
- Money and turnaround. What does it cost (subscription plus training plus disruption), and after how long is that paid back?

The most important word in there is baseline. Without knowing where you started, you cannot prove improvement, and you cannot show a skeptical leadership team or accountant anything. The thread in every success story is also simple: there was someone who owned the outcome, not just the rollout.

A few realistic examples, translated to the owner's level:

Type of business	First pilot (high impact, good feasibility)	What you measure in 90 days
HVAC or construction	Drafting quotes and invoices faster from a fixed line	Hours per week, quote turnaround, error rate
Online store or retail	First-line customer questions (delivery, returns, status)	Share of questions without a human, response time, satisfaction
Professional services	Meeting notes, action items, and follow-up emails	Hours per week per employee, follow-up speed
Clinic or practice	Preparing admin and correspondence	Time per file, wait time for a reply

Watch the numbers you see in the wild (twenty to thirty hours a week, hundreds of percent return). Sometimes they are real, often they are a vendor's best cases. Calculate with your own baseline, not the brochure.

Why it goes wrong (and how you sidestep it)

If 95 percent stalls, the question is not whether it goes wrong but where. Here are the recurring causes, translated from the MIT study and practice, with the counter-move:

- No baseline. You cannot prove a return on something you did not measure beforehand. Counter: measure the current situation first, then the tool.
- No owner. A tool without someone who owns the outcome fizzles out. Counter: name one person responsible for the result, not for the technology.
- Treating AI as plug-and-play. Companies expect instant results without adjusting the way they work. Counter: count on an adjustment period and redesign the process around it.
- AI as a button instead of a fix for a bottleneck. "Let's add AI search" is not a goal. Counter: start with a concrete bottleneck and solve that.
- Messy data. AI does not fix your messy data, it makes the mess louder (sharply put by a B2B practitioner online). Counter: clean up the data the pilot touches, not your whole house.
- The trap of scaling too early. One pilot works, management wants it everywhere, and suddenly you have six half-finished projects instead of one proven one. Counter: go deep on one first, prove it, and only then expand.
- Resistance from your people. If the team does not trust or understand it, nothing happens. Counter: see the next block.

Buy, build, or partner

A striking outcome from the MIT study: companies that use an existing, learning system from a vendor or partner have roughly twice the chance of success as companies that build everything themselves (near two in three versus one in three). For a small business the lesson is clear: start with what already exists.

- Buy or use first. A generalist assistant (Claude or a comparable tool, around twenty dollars a month) covers most first use cases: writing, email, quotes, research, notes. Master one tool, build ten to fifteen standing prompts or Skills in the first two weeks, and only then add a specialized tool for your biggest specific task.
- Partner for the connections. Integrating with your own systems (chapter 14) is where it gets hard; there a partner who knows the path pays off.
- Build yourself last. Only when a process is proven valuable and nothing standard fits do you have something custom built (chapter 13).

And a counterintuitive lesson from the same corner: do not chase away all the friction. Friction (the moments where the AI stumbles or is unsure) is exactly where you learn what the process needs and where a human has to stay. An AI that admits it is not sure is worth more than one that always confidently produces nonsense.

Getting your people on board

This is where most efforts die, not on the technology. The numbers behind failed rollouts keep pointing to people: no internal owner, too little training, poor fit with the real work, and resistance. What works:

- Frame it as help, not replacement. Frame AI as removing boring work, so people keep time for the work that needs judgment and contact. That is not a marketing line; it is how you win trust. (Most owners who use it say AI augments their people rather than replacing them.)
- Start with one power user, not everyone at once. Let one enthusiastic colleague explore it and show it internally. A common barrier is "too little technical knowledge"; one good guide in-house largely solves that.
- Acknowledge shadow AI. Your people probably already use ChatGPT or Claude, even without a policy. That is no reason to panic, but a reason to take charge: make it policy (chapter 8) and point to the safe, business way instead of banning it, because banning drives it underground (chapter 7).
- Make it low-threshold and even fun. Let people use AI on an innocent task first (planning an outing, a birthday) so they feel how it works; then they see the work variants on their own.
- Build a feedback loop in the first 90 days. Let people easily report wrong outcomes. That is how it improves, and how the team feels like an owner instead of a guinea pig.
- Invest in AI literacy. The single biggest predictor of whether adoption succeeds is whether your people understand what they are doing (owners consistently say more training and resources would help most). That is not a burden; it is what makes adoption work.

Governance is not a brake, it is your accelerator

It sounds contradictory, but the companies that scale fastest are the ones with the clearest rules. Governance takes the fear out of the organization, and fear is what slows adoption. You already hold most of this from earlier chapters; here it is as a checklist for leadership:

- Data layer in order. Sensitive and customer data only in an environment that does not train on it (chapter 7). Know which tier your people use.
- Human on the button. For anything sent, deleted, or paid, a human approves (chapters 8 and 14).
- AI inventory and one-page policy. Which tools, for what purpose, with what limits, who owns it (chapter 8).
- Measure on the P&L. Tie every pilot to one number leadership already cares about (turnaround, cost per case, backlog).

Do not see it as bureaucracy but as the seatbelt that lets you drive faster. Without safeguards every pilot stays a demo; with them it becomes a performance accelerator.

A calculation: what is a saved hour worth?

Make it concrete, because leadership decides on numbers. An example:

Item	Assumption	Per year
Time saved	4 employees, 2 hours per week each	over 400 hours
Value of that time	loaded hourly rate €50	about €20,000
Tool cost	4 subscriptions at €20 per month	about €1,000
Setup cost	one-time 2 days to set up and train	one-time

Even calculated conservatively, something like this pays back within a few weeks. But two honest caveats belong with it. Saved hours are only real money if you also spend them somewhere (more customers, less overtime, faster delivery); otherwise it is a feeling of gain. And calculate with your own, measured numbers, not these; this example mainly shows the shape: recovered capacity, set against a small fixed cost, with a short payback. Have Claude run that math with your numbers.

What AI does not solve for you

An honest word, because it is part of good leadership and it builds trust. AI is an amplifier, not a magician. It makes a good process faster, and a messy process faster messy. It does not set your strategy, it does not fix a culture problem, and it does not replace your judgment on decisions that really matter. Where people have to trust each other, where nuance, ethics, and responsibility are at play, the human stays in charge.

See it as an exceptionally fast, tireless colleague who takes everything literally: brilliant in execution, but without its own compass. You set the direction. That is not a limitation to worry about; it is exactly why AI works so well in the hands of an owner who knows what they want. The five percent that succeeds does not do more with AI than the rest, they steer it more sharply.

Pro tips for leadership

- Pick one process per quarter, not five. Depth beats breadth, every time.
- Require a baseline before a pilot starts; no measurement, no pilot.
- Name an owner of the outcome, with a number and a date.
- Reserve a small, fixed budget (rule of thumb: around one to two percent of monthly revenue) and only scale after proof.
- Have your people "build in the open": share internally what works and what does not, that speeds up trust.
- Schedule an explicit moment after 90 days: continue, adjust, or stop, based on numbers.

Common mistakes and how to avoid them

- Buying a platform before you have a problem. Fix: start with the most painful process, not the tool.
- Rolling everything out at once. Fix: one process, one owner, 90 days, then decide.
- Claiming a return with no baseline. Fix: measure up front, otherwise it is a feeling, not proof.
- Selling AI as headcount cuts. Fix: frame it as capacity and quality; that removes the resistance.
- Ignoring or banning shadow AI. Fix: make policy and point to the safe path.
- Scaling too early after one nice result. Fix: depth and proof on one use case first.

Ready to use: an AI opportunity scan for your own business

Below is a filled-in instruction; the situation is already filled in for you, replace the italic parts with yours.

```
You are my down-to-earth AI advisor. I own an HVAC company with 12 employees
[FILL IN: your business, size, sector]. Our biggest time sinks are drafting
quotes, chasing invoices, and answering email [FILL IN: your 3-5 processes].

Do the following:
1. Score each process on Impact (time/money/missed revenue) and Feasibility
   (repeatable, data in order, clear rules), on a scale of 1-5.
2. Put them in a table, sorted by combined score.
3. Pick the best first pilot and explain in two sentences why.
4. Propose a 90-day pilot with: a baseline, 3 measurable goals, who the owner
   is, and what we will NOT do to keep focus.
5. Name the two biggest risks and how we cover them.

Ask me the questions you need before making assumptions.
```

Ready to use: the 90-day review for your leadership team

```
We are 90 days into our AI pilot around [FILL IN: drafting quotes faster].
Here are the numbers: baseline was an average of 45 minutes per quote, now
18 minutes; quality stayed the same; cost €60 per month plus 6 hours of
training [FILL IN: your numbers].

Make a concise one-page decision memo for my leadership team:
- What was the pilot and which number had to go up or down?
- What are the results against the baseline, honestly, including what fell short?
- Continue, adjust, or stop, with reasoning?
- If continue: what is the next step and who is the owner?

Write businesslike and without hype, in plain language for non-technical readers.
```

Do this now

Pick one process today that costs your team too much time and whose rules are clear. Write down how long it takes now (your baseline), name one owner, and have the opportunity scan above draft a 90-day pilot. Agree with your leadership team that after those 90 days you decide on numbers, not on a feeling. You do not have to "do AI"; you have to pick one process and measure it.

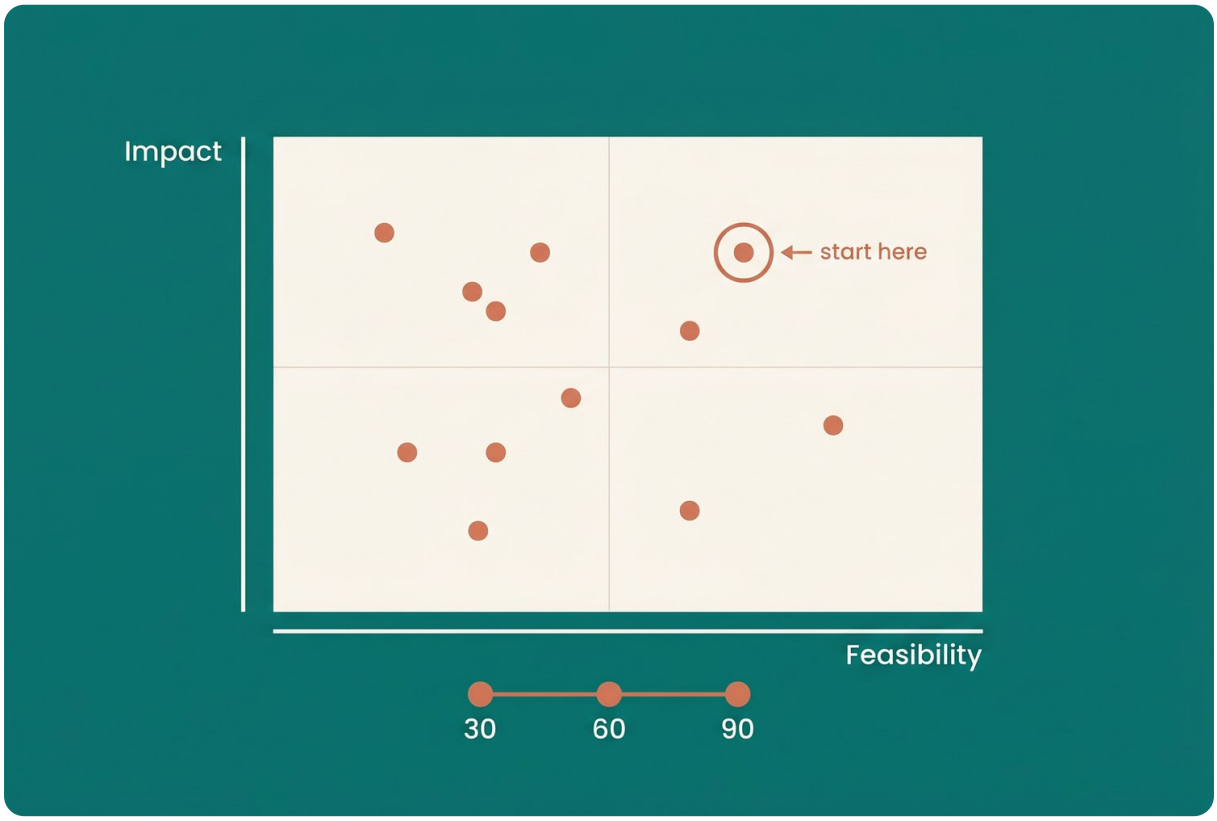


Figure - a simple two-axis matrix (Impact versus Feasibility) with a handful of process dots, and o

In the next chapters we zoom in per department, starting with sales and customer contact: lead research, quotes, your CRM, and follow-up. There the strategy of this chapter becomes concrete work on the floor.

Sales and Customer Contact

If there is one place where AI pays for itself fastest, it is here. Sales is the bloodstream of your business, and here is the good news: the biggest win isn't something spectacular, it's two boring things you're probably leaving on the table right now. Responding first, and following up consistently. This chapter walks the whole line: lead research, the conversation, the quote, your CRM, the follow-up, and the customer contact after that. With honest caveats, because in sales AI can shoot you in the foot just as easily as it can move you forward.

The number that should keep you up at night

The average B2B company takes more than forty hours to respond to a new lead (HubSpot and the Optifai study of 939 companies both land around 42 to 47 hours). Only about a quarter respond within five minutes, and more than forty percent take longer than a day. The bitter figure: roughly 71 percent of B2B leads never get a reply at all (Forbes).

Why that matters:

- Respond within five minutes and you're about 21 times more likely to qualify the lead than if you call after half an hour.
- 78 percent of B2B buyers buy from the company that responds first (Vendasta).
- Responding within a minute can lift conversion by several hundred percent; after five minutes the odds collapse.

Different studies cite different numbers (21x, 100x, a few hundred percent), but the direction is always the same: speed wins. And this is exactly where AI excels. A human can't confirm, route, and give a first relevant answer to every incoming request within a minute. A well-built AI flow can, as long as a human takes over the substantive follow-up. That's your first and biggest sales win: simply being there first.

The first to respond often wins already.

Lead research: from thirty browser tabs to one briefing

Before you talk to someone, you want to know who you're dealing with. That used to mean thirty browser tabs: the website, state business filings, the news, LinkedIn. Now you ask once and get a clean briefing back: what this company does, what's going on, who your contact is, and what hooks you have for a relevant conversation.

The win isn't only time. It's that you stop sounding generic. And that's exactly the bar you have to clear in 2026, because the recipient is more skeptical than ever (see the next section). Good AI lead research gives you the context to show you did your homework.

A few things to keep sharp:

- Work with sources. Ask AI to search and name the source, so you can check facts before you use them in a conversation (chapters 2 and 9).
- Separate fact from guess. Have AI flag what it knows for certain and what's an assumption. A wrong "I read that you were just acquired" is worse than no remark at all.
- Stick to what you may use publicly. No shady data brokering; work with public information and what the prospect shares themselves.

The honest truth about AI outreach

Here I have to warn you, because this is where many companies let AI turn on them. The inbox of 2026 is a battlefield. More than forty percent of all cold email is now AI-generated, and buyers have developed a "delete reflex": about 73 percent immediately trash a robotic, template-shaped email. Worse: 45 percent of buyers are less likely to consider a vendor if the first approach feels synthetic (Chris Rack at B2BMX 2026). "Everyone went fake" is the sentiment.

And the sting is in the technology itself. Gmail's and Outlook's spam filters now judge relevance and engagement, not just sender reputation. A personal, relevant email reaches the inbox, even from a new domain. A generic mass email gets filtered, no matter how cleanly your server is set up. In other words: the system rewards AI that helps you research and be relevant, and punishes AI that helps you send at volume.

The rules that follow from this:

- Use AI for research and relevance, never for volume. Filling in a first name and company name is no longer personalization; it's the absolute minimum everyone already does. Show why you're reaching out to this specific person right now.
- Work human-and-AI together. Let AI research and draft a first version, then put your own voice and judgment over it. In practice the combination delivers two to three times higher reply rates than pure AI.
- Don't sound like the other ninety percent. If your message follows the same pattern as every other cold email, it gets recognized, labeled, and deleted in a fraction of a second.
- Respect the rules. In the EU, the GDPR and the ePrivacy rules require a lawful basis to email people, honest content, and an easy way to opt out; for B2B cold email, legitimate interest can apply, but only with genuine relevance and a clear opt-out. On top of that, since 2024 Gmail and Yahoo reject mail from domains without proper authentication (SPF, DKIM, DMARC). This isn't a detail; it decides whether your message even arrives. (Selling into the US instead? There it's the CAN-SPAM Act rather than the GDPR; the US edition covers it.)

Honest stays honest: cold email still works and delivers strong returns when it's done well, and most B2B buyers still prefer email as a channel. But the spray-and-pray era is over. AI only widens the gap between the two worlds: the generic world sinks further, the relevant world wins.

Quotes that are correct and out the door faster

Building quotes is a silent time-killer for many small businesses. AI can take over most of the drafting work: it pulls the right copy, prices, and terms, fills them in based on what the customer asked, and delivers a draft you review and polish.

If you run SAP Sales Cloud, this is no longer a future promise. Since the Q1 2026 release, the AI reads incoming customer emails (an order request, a confirmation, a buying signal), automatically turns them into a draft sales order with the right items and quantities, and stages it for one-click approval (SAP calls this AI Email-to-Order). That removes the re-keying step that costs so much time in B2B, where customers still often order by email.

The standing rules apply here too: a human approves the quote before it goes out, and sensitive pricing belongs in an environment that respects your data (chapters 7 and 14). AI speeds up the drafting; you stay responsible for what you send.

Your CRM: from data-entry hell to a decision layer

Ask any salesperson and you'll hear it: the CRM feels like data-entry hell. That's where the biggest shift sits. A modern, AI-driven CRM logs activities on its own, proposes the next step, and tidies your pipeline without you having to babysit it. Good salespeople stop using it as a database and start using it as a decision layer: it nudges you toward the right action based on what's happening in the deal ("this prospect viewed your pricing page three times, call today").

Two things make or break that:

- Clean data. AI doesn't fix your messy records, it makes the mess louder (chapter 15). A classic trap is the integration where fields don't match ("Company Size" versus "Number of Employees"), so data doesn't carry over. Let AI help clean up: spot duplicate records, flag incomplete fields, suggest missing data.
- Scoring with a why. A good lead score doesn't just say "82 out of 100," but why it's high and what the best next step is. Let scores drop again on silence, and calibrate your thresholds against your own history (which score actually led to a deal?).

For anyone running SAP Sales Cloud or Service Cloud V2, there's now concretely more. Beyond Email-to-Order, there's a predicted close date for every opportunity, based on real deal data (deal size, velocity through the stages, activity, competition, and seasonality) instead of optimism or management pressure. And there's activity-based risk scoring that flags deals at risk of stalling. The Joule copilot and the ready-made Joule agents work on top of a trusted data layer (the SAP Knowledge Graph and Business Data Cloud) and can sharply shorten multi-step work. A telling example: an agent that combines customer history, support tickets, and the tone in emails to see which customer is about to churn, and hands the account manager a retention offer with customer value attached right away. Since the partnership with Anthropic, Claude can be the reasoning and agent layer there (chapter 14), connected via MCP.

From data-entry hell to a system that nudges you toward the right action.

The crème de la crème: let the agent fill in your CRM

If I could enlarge one thing from this chapter, it's this. Ask your salespeople what they hate and the answer is almost always the same: updating the CRM. The numbers prove them right. About 32 percent of salespeople spend more than an hour a day on manual entry instead of selling, and across the whole day 60 to 70 percent of their time goes to non-selling (Salesforce, State of Sales). No wonder more than half of CRM rollouts fail, with the same number-one cause every time: users who don't keep it up to date.

Run the math. A salesperson typing five hours a week instead of calling costs you not only those hours, but also the deal they didn't follow up because they were typing. For a team of ten, that "CRM tax" quickly runs into six figures a year (a calculation DevRev makes well). And the data that goes in reluctantly often isn't even correct: three quarters of users say less than half of their CRM data is accurate.

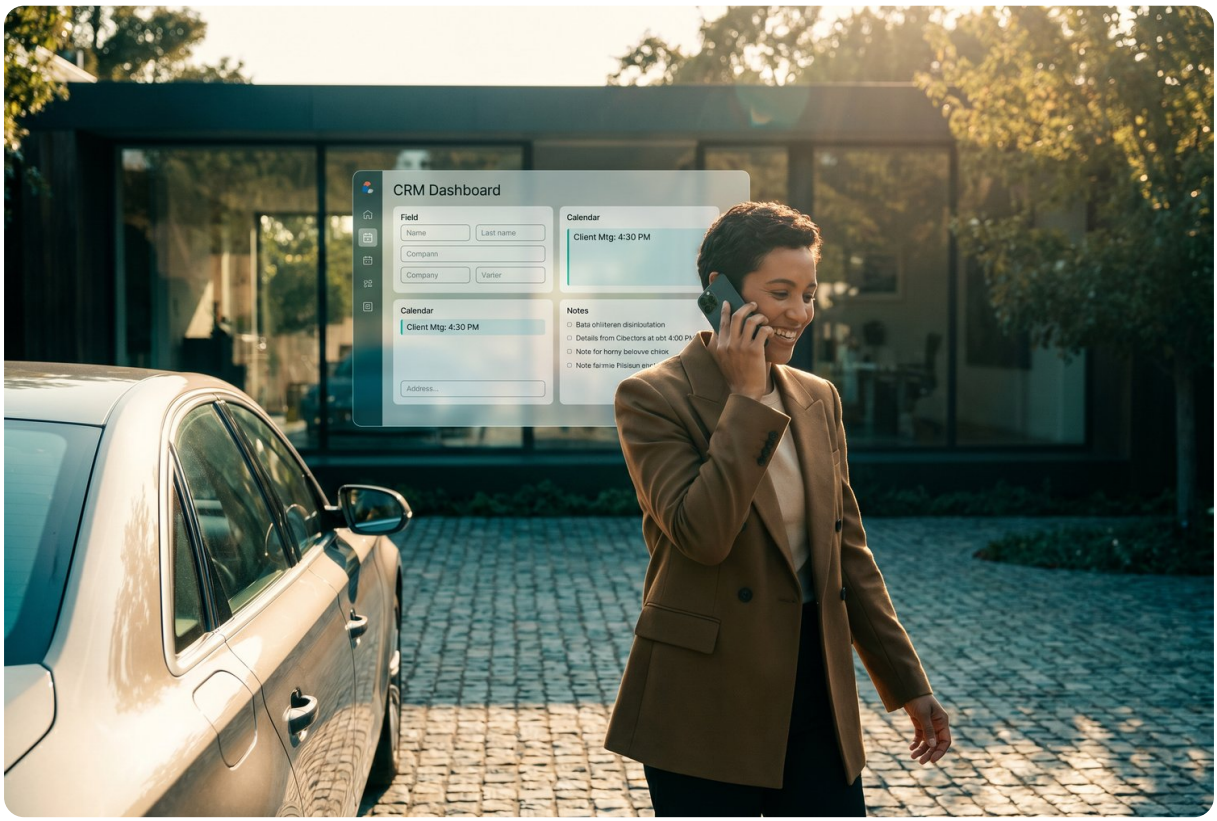
Now the turnaround, and this is what most people don't know yet. You don't have to make your salespeople type at all. You let the AI agent fill in the CRM. After an appointment the salesperson just says: "Just visited Mrs. de Vries, she wants a follow-up Thursday and is interested in the premium package. Set up the appointment, log my notes, and enrich her company details." And it happens. No fields, no tabs, no "I'll do it later" that never comes.

Here's how you set that up, in four steps, and this is exactly the approach that makes the difference:

- Connect or build an MCP for your CRM. Since early 2026, HubSpot has its own ready-made MCP connection that lets Claude read and update contacts, deals, and notes directly in plain language, with no custom code. If you run SAP, it goes through MCP and the Joule agents; for Salesforce through Agentforce or a partner connection. If your CRM isn't covered, you build your own MCP (chapter 14).
- Wrap it in a Skill with your rules. This is the step that really makes it powerful. Build a Skill (chapter 11, with the skill-creator) that knows your quote template, knows which fields are mandatory, what counts as a "qualified lead" at your company, the tone you write in, and how you enrich. The MCP is the hands; the Skill is the craft.
- Share that Skill with your whole sales team. A Skill is transferable. Build it well once, and every salesperson gets the same expert workflow in their hands. Your best sales process becomes the standard process, for everyone, including the new and the less tech-savvy.
- Let your people work in plain language. "Create a quote for customer X based on this conversation." "Enrich this contact with company info and the right point of contact." "Put a follow-up appointment Thursday in the system and log the notes." The agent does the entry; the salesperson checks and moves on.

What this delivers is more than time saved. Notes grow from two half-sentences into full context, because they no longer have to come from a tired memory. The data is correct, because it's captured immediately instead of reconstructed Friday afternoon. And your salespeople get hours a week back to do what they're good at. This is, honestly, the nicest thing you can give your sales team, and the real death blow to traditional CRM entry.

The limits stay hard, precisely because the agent now writes into your system (chapter 14). Start the first week read-only and add write permissions one at a time as you build trust. The agent works under the salesperson's permissions, never with an all-powerful key. And everything that goes out the door, a sent quote or an external email, passes a human. That's how you get the speed without the risk.



You talk, the agent fills it in.

Ready to use: a CRM-entry Skill for your sales team

Below is a filled-in Skill you can copy as text and adapt; replace the italicized parts with your own. Build it out further with the skill-creator if you like (chapter 11).

```
---
name: crm-sales-assistant
description: After a customer contact, processes everything into the CRM.
  Creates or updates the contact and the deal, enriches company data, logs
  appointments, and drafts a quote in our house style. Use this after every
  customer conversation and every incoming request.
---

# Role
You are the sales assistant for [FILL IN: Van Dijk Climate].
You work in our CRM through the connected MCP and follow our rules.

# What you do
1. Contact and company: find the contact; if it doesn't exist, create it.
   Enrich with company size, industry, and the right point of contact from
   public sources, and flag what's an assumption.
2. Deal: update the correct stage and note the next step with a date.
3. Appointment: put follow-ups on the calendar and link them to the deal.
4. Notes: write a 3-5 sentence summary with pain points, budget, and
   objections. No loose fragments.
5. Quote: draft one using our template and our price list.

# Mandatory fields (do not skip)
[FILL IN: lead source, expected revenue, decision date, known competitor]

# Rules
- Send nothing externally without my sign-off; deliver a quote as a draft.
- Work only within the logged-in salesperson's permissions.
- Unsure about a field? Ask; never make something up.
- Tone: down-to-earth and concrete, no sales speak.
```

Follow-up that doesn't stop after 1.3 attempts

This is probably your biggest untapped win. The average salesperson gives up after 1.3 attempts. Winners make six to eight in the first 48 hours. AI helps you keep that up without it becoming work: it proposes a cadence, writes drafts that reference the original request, and reminds you at the right moment.

The art is to keep it personal. Have AI base the follow-up on behavior, not just on a clock: a prospect who opened your quote deserves a different message than one who stayed silent. And let AI make the drafts, but send them as a human, with a last human look. Speed plus persistence plus relevance, that's the formula, and all three are exactly what AI makes easy for you.

The conversation itself: prep, objections, summary

A good sales conversation starts with preparation, and that's exactly where an hour often evaporates. It can be done in five minutes. Have AI pull the insights from earlier conversations with this customer beforehand, name the likely objections, and propose a few sharp discovery questions. Top performers ask eight to ten good questions in a discovery call; AI helps you sharpen them instead of slipping into a pitch.

A particularly useful helper is an objection playbook. Have AI predict the most likely objections ahead of time, based on company size, current tools, the industry, and who's at the table, with a short response frame for each. Pay extra attention to the toughest objection of all: "we're happy with how things are now." About 38 percent of lost deals go to that status quo, not to a competitor.

After the conversation the win is double. Turn a recording or your notes into a summary with action items and a CRM-ready note. The question "what did the customer say about price again?" you then answer by searching, not by guessing. Two caveats: tell the other person if you're recording, and if you use live AI tips during a call, calibrate them to only the important moments. Constant listen-along prompts feel like micromanagement and pull you out of the conversation.

Customer contact and service: answers that are grounded

After the sale, customer contact begins, and there a golden rule applies that has everything to do with trust: an answer must be grounded. A good service AI gives the last confirmed status, with a timestamp and a clear next step. And if the system can't name a confirmed event with a time, it makes no firm claim (a principle from a 2026 customer-service playbook). That's the same lesson as hallucination in chapter 2: better an honest "I'm not sure about this, I'll find out" than a confidently announced error.

Draw a hard line around that: anything touching money (a refund, a credit, a commitment) passes a human. The companies that do this well have set up fixed checkpoints precisely there. If you run SAP Service Cloud, you see this same pattern in the service agents: lightning-fast draft replies and triage, with the human on the button for what matters. That's how customer contact gets faster without becoming impersonal or unreliable.

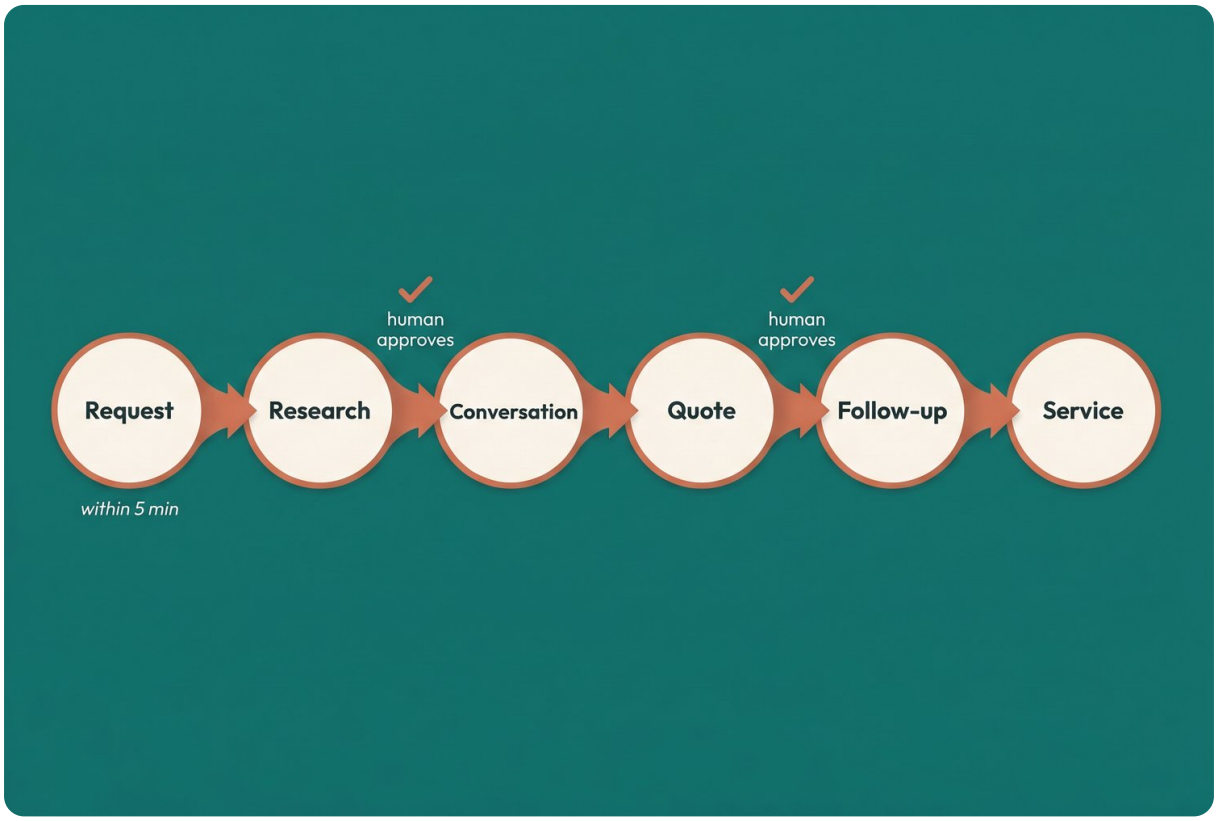


Figure - the sales and service flow as one line: request (respond within 5 min) -> research -> conv

What you deliberately don't automate

With all that speed it's tempting to automate everything. Don't, because in sales the relationship is the product. A few things you deliberately keep human.

The core of relationship-driven selling stays yours. If you sell mostly to existing customers and partners, AI helps you with prep, presentations, and looking things up, but not with building trust. Match the effort to how you sell: a "hunter" with hundreds of cold accounts gets more out of automation than a "farmer" who deepens a handful of relationships.

Precisely now that buyers are flooded with synthetic messages, a real human touchpoint becomes a differentiator instead of a cost. Whoever is the only one to send a genuine, relevant message or just pick up the phone stands out among the automated noise. Also consider what a channel is for: email is a work tool between people who already know each other and want to get something done, while discovering and orienting happens more on LinkedIn, through a referral, or at an event. So don't expect cold email to do the relationship work for you.

And the judgment moments stay yours: giving room on price, sensing the mood at the table, having the hard conversation. Let AI speed up the routine work, so you actually have more human time left for the moments that truly make a deal. That's not a brake on AI; that's how you use it well.

Pro tips from practice

- Make speed your first project: ensure every new request gets a confirmation within minutes, even outside business hours.
- Use AI to research and personalize, not to spam; the inbox rewards relevance and punishes volume.
- Let AI prep your conversation in five minutes: previous contacts, likely objections, three sharp questions.
- Record conversations (with consent) and have them summarized; your follow-up gets sharper and you no longer have to remember anything.
- Keep your CRM clean; an AI on messy data mostly amplifies the mess.
- Send nothing unseen: AI writes the draft, you put your name under it.

Common mistakes and how to avoid them

- Responding to leads too late. Fix: automate the first confirmation and routing, then take over as a human.
- Using AI for mass outreach. Fix: use it for research and relevance; personalize for real or don't do it.
- Selling first-name personalization as "tailored." Fix: show why this specific person, with real context.
- Giving up on follow-up too early. Fix: a cadence of six to eight touchpoints, each referencing the request.
- Leaving dirty CRM data in place. Fix: have AI clean up (duplicate, incomplete, mismatched fields) before you build on it.
- Letting the service AI make firm claims. Fix: only grounded answers with status and time; money matters pass a human.

Ready to use: a five-minute conversation prep

Below is a filled-in prompt; the situation is already filled in for you, replace the italicized parts with your own.

```
You are my sales assistant. Tomorrow I have an intro meeting with the
purchasing manager of a mid-sized furniture manufacturer [FILL IN: company,
role, industry]. We provide maintenance contracts for production machinery
[FILL IN: what you sell].

Prep me:
1. Give a short profile of the company based on public sources, with source
   attribution, and flag what's certain and what's an assumption.
2. Name 3 possible reasons why maintenance might be on their mind now for
   this type of company.
3. Give 5 sharp, open discovery questions (no pitch).
4. Predict the 3 most likely objections, including the status quo ("we
   handle this ourselves"), with a 2-sentence response frame for each.
5. Propose 1 concrete next step to suggest at the end.

Ask me the questions you need before making assumptions.
```

Ready to use: a follow-up email that doesn't sound generic

```
Write a follow-up email for a prospect who opened our quote for a maintenance
contract last week but didn't respond [FILL IN: situation]. Context I want to
use: in our conversation she mentioned that a breakdown last year cost three
days of production downtime [FILL IN: a real detail from your contact].

Requirements:
- Reference that detail concretely, no generic "just checking in."
- Max 120 words, in my down-to-earth tone, no sales speak.
- End with a low-friction question, not "when can we talk."
- Give me 2 versions so I can choose.

I'll send it myself after a final check.
```

Do this now

Take your biggest sales leak: probably your response time or your follow-up. Set up today so that every new request gets a first, relevant response within minutes, and for your open quotes set up a follow-up cadence with AI drafts that you review. Use the conversation prep above for your next appointment. Getting one thing right (being fast, or staying on follow-up) often delivers more than ten tools at once.

In the next chapter we move to marketing and content: campaigns, your content

Marketing and Content

Marketing is where most small businesses take their first steps with AI, and where most also hit their first disappointment. You can now make a lot of content faster than ever, and that's exactly the trap. More content isn't the goal; the goal is that your best ideas travel further, that they sound like you, and that you're the answer when a customer asks an AI. This chapter is about how to use AI for marketing without disappearing into the gray mass, and how to make images and video in a way that was unthinkable a few years ago.

Let's start honest. On marketing forums in 2026 you hear the same sigh over and over: "I produce way more now, but my engagement is flat and my conversion barely moves." That's no coincidence. If everything you make looks like everything everyone else makes, it disappears. The way out isn't less AI, but smarter: voice first, production second.

Don't make more. Make your best ideas travel further.

Voice first, output second

The biggest worry of anyone using AI for content is always the same: that it sounds generic, that everyone sees through it instantly. Fair. And the solution is surprisingly concrete, because there's a misunderstanding here. AI doesn't learn your voice from adjectives. "Write professionally" gets you nothing. AI learns from examples.

Here's how you train your brand voice, in a few steps:

- Name five to seven concrete traits of your voice, and say what they mean. Not "professional," but "down-to-earth, concrete, with short sentences and the occasional dry joke."
- Give five to ten real pieces that represent you well: your best posts, emails, a page from your site. Have Claude analyze them ("summarize my writing style and tone") and compare that to what you intended.
- Capture it as a reusable voice profile of a few lines that you include in every prompt, or better: set it as a project instruction or a Skill (chapters 10 and 11), so every output leans on it automatically.
- Work with counter-examples: show what specifically doesn't fit you, including words you never want to use because they reek too much of AI.
- Iterate. Never just accept the first version; ask "this is too formal, loosen it up" until it's right.

The best test is an honest question you ask of every piece: would I read this myself, do I get it immediately, and could any random other company have written this too? If the answer to that last one is "yes," you have work to do. This is exactly what sets your brand apart, and it's built in a handful of hours.

One good idea, twenty pieces: content atomization

Here lies the biggest lever in your content production, and it's called atomization. The idea, once coined by Todd Defren and made famous by Gary Vaynerchuk's team that turned a single talk into hundreds of social pieces: you take one large, proven piece of content and break it into dozens of smaller ones, each made for its own channel. No copy-paste; you build something that belongs on each platform.

Here's how it works, as a hub and spokes:

- Pick a proven core piece (the hub). A long-form guide, a webinar, a research study, a customer case, a podcast episode. Choose based on what already did well; atomizing a vague idea only multiplies the noise.
- Pull out the separate components: a strong statistic, a sharp quote, a step-by-step, a framework, a surprising insight, a chart.
- Reassemble each component into a platform-native form. The same insight becomes a thoughtful post with context on LinkedIn, a visual card with a practical tip on Instagram, a short block in your newsletter.
- Spread it across your calendar and keep pointing back to the core piece. One investment of a few hours delivers weeks of content, with the cost per piece collapsing.

Below is a filled-in prompt; the situation is already filled in for you, adjust the italicized parts.

```
I have a proven core piece: a 2,000-word guide on heat pump maintenance that
scored well [FILL IN: your best piece]. My channels are LinkedIn, our
newsletter, and Instagram [FILL IN: your channels]. My brand voice:
down-to-earth, concrete, no jargon [FILL IN or reference your voice profile].

Atomize this piece:
1. Pull out the 8 strongest separate components (stat, quote, step, insight).
2. Make 3 platform-native pieces per channel, in my voice, no copy.
3. Propose a publishing order over 3 weeks, each time with a reference back
   to the guide.
4. Flag which 2 pieces have the most potential and why.

Ask me the questions you need first.
```

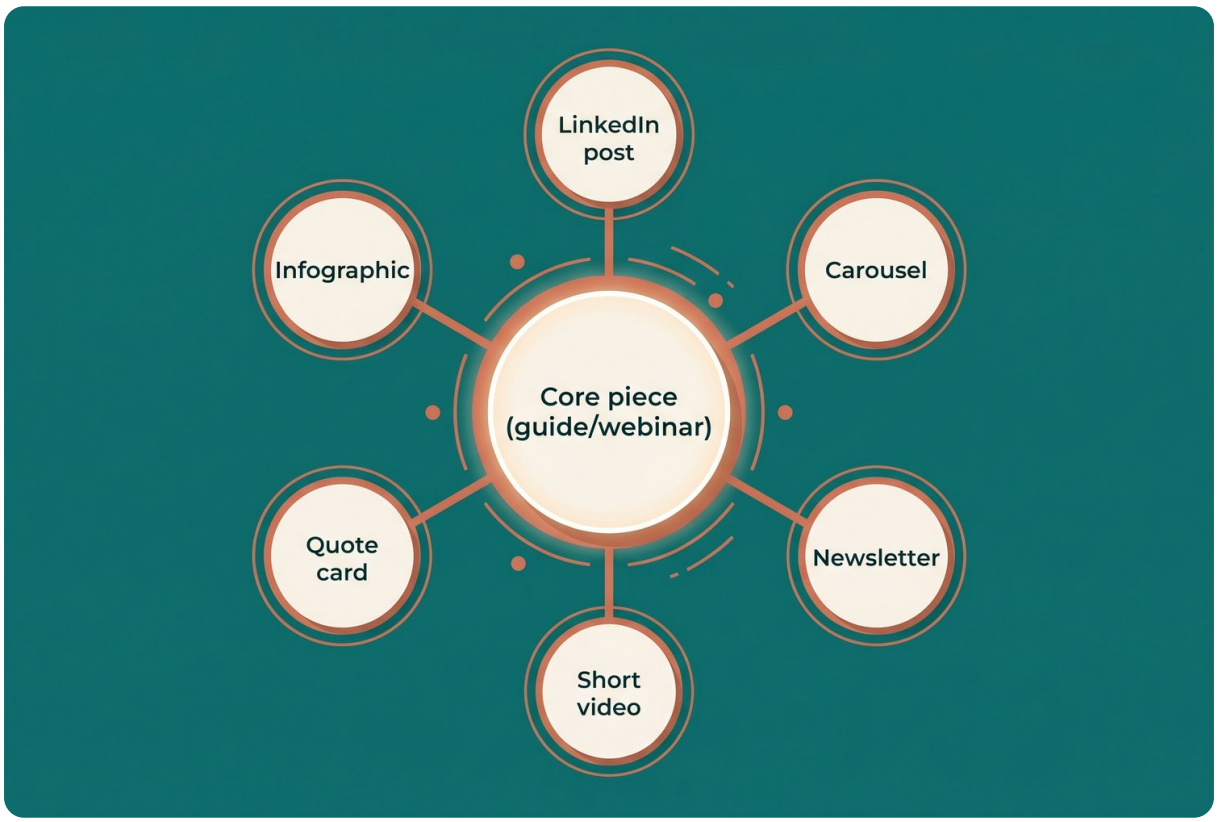


Figure - a hub-and-spokes model

Image and video: creating content was never this easy

Until recently, text was the easy part and visuals the bottleneck. That has flipped. With the right connection, you make images and video straight from your conversation with Claude, no editor, no API keys, no learning one tool after another.

The hinge point is the Higgsfield connection (the MCP from chapter 14). Since late April 2026 it turns Claude into a complete production studio. One connection gives access to more than thirty image and video models at once (from Soul and Flux to Seedream, Kling, and Veo). You describe the shot you want, and Claude picks the right model, sets the duration and aspect ratio, starts the generation, waits until it's done, and delivers the result back into your conversation. Output up to 4K, videos up to fifteen seconds, in any ratio your distribution needs. You pay with your own Higgsfield credits; there's nothing to install.

What that means in practice: you say "make three vertical product heroes with the headline Sleep Cool on them, and four short vertical videos for social," and it happens. Want consistent characters or a fixed style across all your images? Soul training helps with that. And because everything runs through connections, you can stack it: in one prompt, have the images made and staged right into your web store or as an ad variant, provided those connections exist. Three systems, one prompt.

Higgsfield isn't the only one. It's useful to know the landscape, because each model has its strength:

- Image. For production work, Nano Banana Pro (from the Gemini family) is currently the strongest: sharp 4K images, reliable text in the image, and control over light, camera, and composition via plain language. It can even make a full set of product photos for a web store from a single phone snapshot. Midjourney stays king for mood and moodboards, but is weaker when you need exact text or a tight layout.
- Video. Veo and Sora deliver cinematic clips (Veo with sound), Kling is strong and cost-efficient for social, and avatar tools like HeyGen and Synthesia make talking-presenter videos for explainers and training. For short social content, you auto-clip long videos into shorts with a tool like OpusClip.

Two honest caveats, because this is no magic wand. AI is at its best for social posts, explainer videos, product demos, and ad variants; for a true brand story or an emotional commercial, human production still wins on subtlety. And: always review and refine before you publish. The speed tempts you into posting mindlessly, and that's exactly how you end up in the generic mass. Making images has become easy; taste and judgment stay yours.

You describe the shot, the agent makes it.

Advertising with AI: more variants, sharper creative

Anyone advertising on Meta, Google, or TikTok already notices it: the creative is the new target. The ad systems pick the audience, the budget, and the placement themselves; what you supply, the images and the copy, now determines most of the result. And that's exactly where AI moves the limits.

The old bottleneck was creative production: you could scale your budget and audiences fast, but making enough good ads cost time and money. That bottleneck is gone. Where a team used to test five to ten variants, you now make fifty to a hundred, and the ad AI learns faster the more variants it gets to see. Meta's Advantage+ already turns one supplied image into dozens of versions automatically (different copy, ratios, music) and scales the winners on its own. Since early 2026, these improvements are on by default.

A few practical points:

- Supply variation, not isolated gems. Give the AI multiple angles and formats; vertical video (9:16) is the most important format because nearly all traffic is mobile.
- Upload your brand kit. Logos, colors, and fonts then get applied automatically across all AI variants, so scale doesn't come at the cost of your brand.
- Refresh in time. An ad is on average worn out after about three weeks; plan two to four refreshes a month and have AI make new variants from what already works.
- Add UGC. Authentic-looking content (a review, an unboxing, a product-in-use) often performs better in these automated environments than polished brand video.

That last one, UGC, is a chapter of its own. There's a whole generation of tools (Arcads, Creatify, HeyGen) that turn a product page or a script into a video where an AI actor talks about your product, ready for social. You paste a link, and out rolls an ad. It works, but there's one hard law: a viewer who realizes within a fraction of a second that it's an AI figure is already gone. The difference between "plastic" and believable has become the most important performance dial. So use it for what it does well (quick variants, testing cold traffic) and be honest: for a brand story that has to land, a real person or real customer stays stronger.

And two sober caveats. The machine is no miracle cure: Meta's own system, by its own admission, often misses the best-performing creative, so keep testing, review weekly, and keep clean customer data as the base. And mind transparency: a realistic AI video of a "customer" touches the rules from the next section and your audience's trust. Speed is nice, but your brand stays yours.

Social and the honest truth about AI content

It's tempting to put your whole social channel on autopilot. Don't. Practice and research point the same way: automating everything produces generic marketing, and buyers trust AI content less than marketers think. What does work is a clear division of labor.

- Let AI do: the research and ideation, the repurposing, segmentation, drafting concepts, and reporting.
- Keep human: your strategy, your brand voice, the creative direction, and the final check.

In other words: AI is your intern and your production line, not your editor-in-chief. A draft AI makes gets your eye and your voice before it goes into the world. That's how you combine the pace of AI with the recognizability that makes people stop scrolling.

Getting found in three layers: SEO, AEO, and GEO

Getting found is no longer one game but three, and they stack on each other. Whoever plays only the first becomes invisible in 2026; whoever plays all three is everywhere their customer searches.

The first layer is classic SEO: ranking high in Google and Bing with good content, a fast and technically healthy site, and a few strong links. That layer stays the foundation. AI just makes the work much lighter. Have Claude research your keywords and search intent, make a content brief based on the top-ranking pages, sharpen your headings, meta text, and internal links, and run technical audits. What used to take a whole SEO agency, you now kick off with a few prompts.

The second layer is AEO, answer engine optimization: making sure your answer gets picked up as the direct answer, in a Google AI Overview, a featured snippet, or a voice assistant. The approach differs from regular SEO: put a short, direct answer of forty to sixty words at the top of your page, use a question-and-answer structure with FAQ markup, and aim at question keywords (how, what, why).

The third layer is GEO, generative engine optimization: getting named and cited in the answers of ChatGPT, Claude, Gemini, and Perplexity. The trend is unmistakable: Gartner predicts that traditional search engine volume will drop by about 25 percent by 2026 as people shift to AI assistants. Where SEO makes you findable, GEO makes you part of the answer. Princeton's study that coined the term found that a few things sharply raise your visibility in those answers: adding expert citations, quotations, and statistics can lift visibility by up to roughly 40 percent, with clear source attribution.

Important: these layers overlap, but aren't the same. Research from Ahrefs showed that only a small share (around twelve percent) of the pages cited by AI also sat in the Google top ten. So ranking high helps, but it doesn't guarantee a spot in the AI answer. And because AI Overviews now appear in a sizable share of searches and depress click-through significantly, being number one is no longer enough; you want to be the source that gets cited.

Working on GEO concretely, as a small business:

- Do a baseline measurement. Ask ChatGPT, Perplexity, and Gemini the questions your customer asks ("best HVAC contractor for heat pumps near [city]," "what does an AC maintenance plan cost"). Are you in there? How are you described? Who else?
- Build consensus across multiple sources. AI trusts you more when your message comes back consistently on your site, on YouTube, on Reddit, on trade platforms, and in reviews. Contradictory signals lower that trust.
- Get into the sources AI already cites. Reddit and Wikipedia are among the most-cited sources; genuinely taking part in a relevant Reddit discussion (no spam) can lift your visibility faster than you'd think.
- Structure for extraction: the answer up front, clear headings, concrete facts and figures.
- Measure your share of the answers, not just your traffic. A lot of AI search traffic is "zero-click," so your old stats no longer tell the story.

Below is a filled-in prompt for that baseline; adjust the italicized parts.

```
I want to know how visible my company is in AI answers. I'm an HVAC company
in central the Netherlands [FILL IN: your company and region].

Do the following:
1. Write 10 questions my customer would ask ChatGPT or Perplexity, from
   browsing to ready-to-buy.
2. For each question: search the web and describe which kinds of companies
   and sources are likely named, and whether I'm among them.
3. Point out the 3 biggest gaps: where am I invisible while competitors
   show up?
4. Give 1 concrete action per gap (a page, a Reddit contribution, a review
   spot) to close it.

Be honest; don't invent mentions that aren't there.
```

A contrarian but important note to close: traffic from AI answers often converts better than regular search traffic, because a recommendation from an AI feels like advice, not an ad. Whoever gets named gets not just clicks but trust. And an honest warning: no one can guarantee you a spot in an AI answer. Anyone who promises that is selling air.

Transparency and disclosure

One thing belongs here for the EU SME, and it connects to chapter 8. The transparency rules of the EU AI Act (Article 50) start applying on 2 August 2026; the technical requirement to add a machine-readable watermark to AI content has been pushed to 2 December 2026. The core for marketing: if you make a deepfake (a manipulated image, audio, or video that looks real), you must make that clear. And text you publish to inform the public about matters of public interest must be labeled as AI-generated, unless a human has reviewed it and takes editorial responsibility for it.

For most ordinary marketing (a product page, a social post you edit yourself and put your name under), that human review usually keeps you outside the strict labeling duty. But transparency is the safe default, and with realistic AI images or video of real people or events you stay careful anyway. Record it in your AI register: what you make with AI, and where you label it. It costs little and it builds trust. And honest reviews matter too: passing off a fake or AI-generated "customer" as real is deceptive under EU consumer law, so don't do it.

(Selling into the US instead? There it's the FTC's rules on deceptive practices and fake reviews rather than the EU AI Act; the US edition covers that.)

Pro tips from practice

- Build your voice profile first and save it as a Skill or project instruction; after that, all output leans on it.
- Atomize only your proven top performers, not everything; quality travels further than quantity.
- Have image and video made through the Higgsfield connection straight from your conversation; let Claude pick the model unless you have a preference.
- Lock a style or seed for consistency when you need a series of images in the same look.
- Do a GEO baseline every quarter; knowing how AI describes you is half the work.
- In ads, test many variants instead of a few perfect ones; supply your brand kit so the AI stays within your style.
- Publish nothing unseen: AI delivers the draft, you put your name and your taste under it.

Common mistakes and how to avoid them

- Making more instead of better. Fix: fewer pieces, sharper voice, carry proven ideas further.
- Letting AI write without a voice profile. Fix: train your voice with examples and lock it in.
- Putting everything on autopilot. Fix: AI for research and draft, human for strategy, voice, and final check.
- Posting image and video mindlessly. Fix: always review and refine; AI is your studio, not your taste.
- Steering only on Google. Fix: play all three layers, SEO, AEO, and GEO, so you also show up in AI answers.
- Running AI ads on blind autopilot. Fix: supply variation and your brand kit, refresh every few weeks, and review weekly.
- Publishing deepfakes or realistic AI images unmarked. Fix: label them and note it in your AI register.

Ready to use: your brand voice in a profile

Below is a filled-in prompt; adjust the italicized parts and save the result as a Skill or project instruction.

```
I want you to learn my brand voice and summarize it in a reusable profile.
Here are 5 pieces that represent me well: [FILL IN: paste 5 posts/emails/
pages]. My brand is an HVAC company that communicates in a down-to-earth,
honest way [FILL IN: short description].

Do the following:
1. Analyze tone, sentence structure, rhythm, and word choice; summarize my
   voice in 5-7 concrete traits.
2. Name 10 words or phrases I typically do use, and 10 I'd never say
   (including overly AI-ish clichés).
3. Write a voice profile of max 8 lines that I can paste into every prompt.
4. Test it: rewrite this example sentence in my voice and explain what you
   changed: "We provide high-quality, sustainable solutions for your comfort."
```

Ready to use: a visual brief for Higgsfield

```
Through the Higgsfield connection, make image and video for a campaign.
Product: a quiet heat pump, audience homeowners [FILL IN: your product and
audience]. Style: calm, light, Scandinavian, daylight [FILL IN: your look].

Deliver the following:
1. Three vertical images (9:16) with the headline "Quiet. Efficient. Done."
   legible in the image.
2. One square image (1:1) of the product in a real living room.
3. Two short videos of max 12 seconds for social, same style.

Pick the best-fitting models yourself, keep the look consistent, and deliver
the results with a short note per piece. I'll then choose which we use;
nothing goes out without my sign-off.
```

Do this now

Start with your voice, not your output. Paste five of your best pieces into Claude and have it make a voice profile that you save as a Skill. Then take one proven piece of content and have it atomized to your two most important channels. And connect Higgsfield (chapter 14) so you make your next social images straight from your conversation. One voice, a few proven ideas, and image on demand: that beats ten competitors who only post more.

In the next chapter we move to operations, admin, and finance: SOPs, invoicing, data analysis, and reporting. There, AI doesn't make your story prettier, but your business tighter.

Operations, Admin, and Finance

This is the chapter of the boring hours, and with that the chapter with the biggest win. Chasing quotes, re-keying invoices, closing the month, throwing a report together: it's work no one misses and yet it eats your week. AI takes most of that out. But it's also the chapter where you have to be most careful, because here AI touches your money and your numbers. The through-line: let AI do the mechanical work, and put a human on the button for anything that moves money or backs a decision.

So two things are true at once. The time saved is enormous and real. And an error in a number is more expensive than an error in a sentence. This chapter shows how to grab the first without falling into the second.

The boring hours back, without losing control.

SOPs: get the knowledge out of people's heads

Don't start with the numbers, start with your processes. In most small businesses the knowledge sits in the heads of a few people. How you build a quote, how you handle a complaint, exactly how the month-end close goes. When such a person goes on vacation or leaves, the knowledge walks out the door. An SOP, a standard operating procedure on paper, solves that, but no one has time to write them. AI does.

You describe a process in plain language, or you record your screen while you do it once, and AI turns it into a clean, structured procedure: steps, who's responsible for what, points of attention, and a version date. What used to be a months-long documentation project is now an afternoon's work. And the best part: that SOP immediately becomes your onboarding material and your training guide. A new hire no longer reads over a shoulder, but follows a procedure that's correct.

One honest caveat. AI makes SOPs fast, but sometimes too long and too generic. Let it help, not have the last word: an experienced colleague reviews the procedure for what actually matters in practice. An SOP no one trusts doesn't get used.

Operations: planning, inventory, and purchasing

Before we get to the numbers, the shop floor itself. Operations is a chain of planning, ordering, and adjusting, and that's exactly where a lot of time disappears into back-and-forth and gut feeling. AI makes it tighter.

- Demand forecasting and inventory. Have AI read your sales and inventory data and forecast demand, so you order in time and don't run out or get stuck with dead stock. In practice, forecast errors drop by twenty to fifty percent, which translates directly into fewer lost sales and fewer write-offs.
- Purchasing and suppliers. Have supplier quotes laid side by side, terms compared, and an order proposal made that you approve. Some systems (like the order-reliability agents SAP announced for 2026) flag a supply risk, a looming shortage, or a delay before your order suffers.
- Scheduling and rosters. Building a schedule, a route plan, or an appointment calendar based on availability and priority is exactly the kind of puzzle AI quickly makes a good first proposal for, which you then refine.
- Finding bottlenecks. Give AI your process data and ask "where are the bottlenecks?" It sees patterns that stay invisible in a long list: the step where everything stalls, the day it goes wrong, the customer or supplier who keeps causing delays.

The rule holds here too: AI may forecast and propose, but the order that costs money, you approve. That's how you keep the speed of automation with the grip of a human who knows the context.

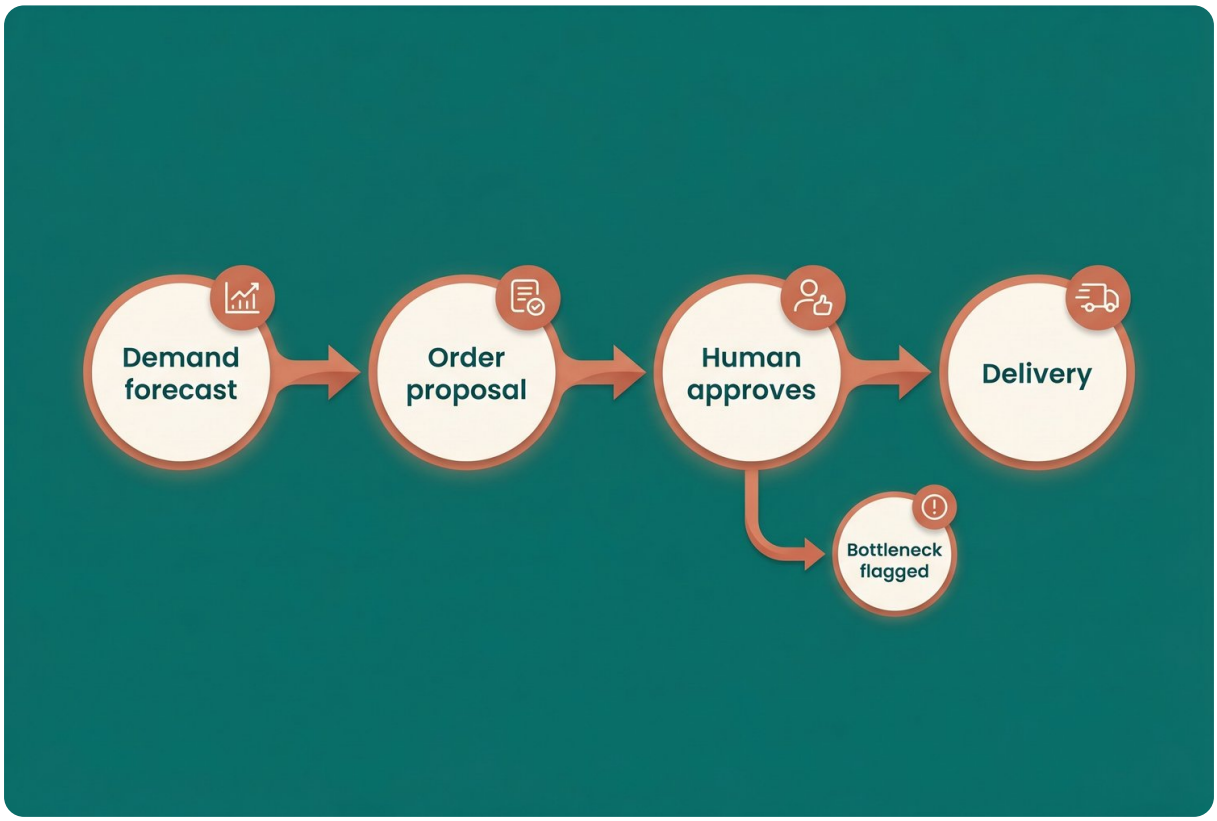


Figure - a simple operations line from demand forecast to order proposal to "human approves" to del

Invoicing and admin without re-keying

For many businesses this is the first, easiest win. Processing an invoice by hand, from arrival to payment, easily costs three quarters of an hour, and finance teams spend about ten days a month on it. Modern AI reads the invoice (from an email, a PDF, or a photo), pulls out the supplier, amounts, and line items, and stages everything in your system. Recognition is now at 95 to 99 percent, where two years ago it was still 70 to 80. What's left is a check of a few minutes instead of re-keying.

What AI concretely does for you here:

- Read and code. Pull data from any invoice form and automatically post it to the right general ledger account.
- Match. Compare the invoice with the purchase order and the receipt (the three-way match), so discrepancies surface before you pay.
- Flag duplicates and fraud. A duplicate invoice or a supplier's suddenly changed bank account number gets flagged.
- Remind and follow up. Chase outstanding invoices with polite reminders, and stage payments for the right date.

Note the pattern the good systems follow: edge cases, like a handwritten invoice or a strange layout, they don't guess at, but put them to a human. That's exactly the disposition you want. For anyone running SAP, this connects to the Email-to-Order from chapter 16: the customer emails, the AI makes the draft, you approve.

The effect, in sober before-and-after terms as you see it in practice:

Task	Before AI	With AI
Bank statement reconciliation	4 to 8 hours per account per month	minutes, human only reviews the exceptions
Process an invoice	3 to 5 minutes, 2 to 4 percent errors	seconds, around 0.5 percent, auto-posted
Month-end close	8 to 15 business days, step by step	4 to 6 days, continuous and parallel
Detecting anomalies	quarterly sample, catches 60 to 70 percent	real-time, catches 95 percent within a day

Your numbers in plain language: data analysis and reporting

This is where it really starts to shift for most owners. You've had data in Excel and in your systems for years, but getting an answer out of it took an analyst or an afternoon of puzzling. That's over.

The biggest leap is Claude in Excel, an add-in broadly available since early 2026 (from the Pro plan up) and part of Cowork. It's not a formula helper; it reads your whole workbook, understands nested formulas and the relationships between tabs, and gives a reference to the exact cell with every answer. You can ask "which assumptions drive the Q3 revenue forecast?" and get an answer you can trace. You can say "raise growth by two percent and show the impact," and it adjusts the assumption while preserving all formulas, with the change clearly marked. It builds and repairs models (from a three-statement financial model to an investment calculation), tracks down error messages, and summarizes a workbook into an executive dashboard. Through connections it even pulls in current external data.

Outside Excel the same principle works: upload an export or a CSV and ask your question in plain language. "Who are my twenty biggest customers and how are they growing?" "Where are the bottlenecks in this process data?" "Compare the actuals to the budget and explain the biggest variances." The analyst in you doesn't go away; the boring collection work does. The pattern that saves finance teams the most time: AI pulls the numbers from three files, names the variances, and writes a draft commentary, and you review, refine, and approve instead of hauling it yourself.

Connect this to the scheduled tasks from chapter 12 and your reporting runs on its own: every Monday morning a fresh sales or cash-flow overview in your folder, ready for your eye.

Your data already had the answer; now you can just ask it.

Looking ahead: forecasting, close, and anomalies

Where AI makes your admin tighter, it makes your steering sharper. A cash-flow forecast used to be a weekly spreadsheet with a margin of plus or minus a quarter beyond thirty days; with a model that learns daily, you sit around ten percent at ninety days. More important than the exact number: the system sees a shortfall coming before it's here, and flags margin loss early.

The same goes for risk. Instead of a quarterly sample that misses most anomalies, a continuous check runs that spots unusual transactions almost immediately: a double payment, a strange entry, a cost item out of line. For a small business, this is the difference between being surprised after the fact and stepping in on time. And your month-end close, which might now take two weeks, you can bring back to a handful of days by automating the reconciliation and the standard postings and only doing the exceptions yourself.

The golden rule with numbers: trust is good, verifying is better

Now the most important thing in this chapter, and the reason finance demands caution. AI sometimes makes things up, including numbers, and does it with full conviction. A longer, more confident, "reasoned" answer isn't automatically more reliable. In a text, an error is annoying; in financial statements or a tax filing, it can cost you money, your reputation, or a penalty. And you stay responsible, not the AI.

So treat every number and every commentary AI delivers as a draft you still have to check. The rules of thumb:

- Trace it to the source. Don't have AI only summarize, but reference the underlying numbers, so you can lay a sample back onto the raw data.
- Keep a log. What did you ask, what did it answer, which version, and what did a human change and why. That's your audit trail if something later doesn't add up.
- Work by exception. Have AI handle the routine and only surface the edge cases, sorted by amount and risk. That way your attention goes to what matters.
- Put a human on the button for money. A payment, a filing, a commitment: a human approves that. AI checks the numbers, but it can't judge intent.
- Sensitive financial data belongs in an environment that respects you (chapter 7), and your workflow belongs in your AI register (chapter 8).

An honest example from practice: an experienced user had Claude work in a complex twenty-tab model, with a detailed instruction attached. It helped well, but still deleted a formula it should have left alone. The lesson isn't "don't do it," but "give it good instructions, protect your formula cells, and review every change." With that posture you get the speed without the scare.

Pro tips from practice

- Start with SOPs: capture your most important processes first, so AI (and your team) knows how it's supposed to go.
- Take invoicing as your first finance win; the time saved is immediate and the risk is low if a human reviews the exceptions.
- Use Claude in Excel with a short explanation of your model attached (or a Skill), and protect your formula cells.
- Have reports stand ready on their own every week via scheduled tasks (chapter 12).
- Always ask AI for the source of a number, not just the result.
- Keep a simple log of what AI did in your admin; it's your audit trail.

Common mistakes and how to avoid them

- Letting AI produce a number without a source. Fix: ask for references and lay a sample back onto the raw data.
- Letting a payment or filing go through automatically. Fix: a human approves what moves money.
- Expecting AI to fix messy data. Fix: clean up the data the task touches; otherwise AI amplifies the mess.
- Publishing SOPs mindlessly. Fix: have an experienced colleague review them for practical value.
- Keeping no audit trail. Fix: log prompts, answers, and human corrections.
- Leaving formula cells unprotected during model work. Fix: protect them and review every change.

Ready to use: an SOP from a process description

Below is a filled-in prompt; the situation is already filled in for you, replace the italicized parts with your own.

```
Make a clear SOP (standard operating procedure) for our quoting process at an
HVAC company [FILL IN: your process and company]. Here's how it goes now, in
my words: customer calls, we schedule a site visit, technician takes
measurements, office builds the quote in our system, and we follow up after
three days [FILL IN: your steps].

Deliver the following:
1. A numbered step-by-step with who's responsible per step (RACI-light).
2. Per step, the points of attention and the most common mistakes.
3. The mandatory checks before a quote goes out the door.
4. A version date and a place for the owner.

Keep it short and practical; I'll have an experienced colleague review it after.
```

Ready to use: a monthly report with commentary

```
I'm uploading our numbers for this month (sales, costs, outstanding items)
[FILL IN: your files]. Our budget was at €120,000 in revenue and €85,000 in
costs [FILL IN: your budget].

Do the following:
1. Compare actual to budget and name the three biggest variances, with a
   reference to the underlying numbers so I can trace them.
2. Write a half-page commentary in plain language, honest, including what
   disappointed.
3. Name two notable or unusual entries that I'd better check myself.
4. Give a short outlook for next month based on the trend.

Flag what's certain and what's an assumption. I check and approve myself.
```

Do this now

Pick your biggest administrative time-sink, probably your invoicing or your monthly report. First capture the procedure as a short SOP, so AI and your team know how it's supposed to go. Then have it build the report for one month with the prompt above, and check the result against your raw numbers. If that feels good, put it on a fixed schedule. You win back your week, and you keep control where it counts.

In the next chapter we move to service and support: draft replies, your knowledge base, and gauging customer satisfaction. There, AI shifts your customer contact from reactive to forward-looking, with the same rule: grounded answers, and a human on the button where it matters.

Service and Support

Customer service is where the promise of AI and its bad reputation collide. Everyone has been stuck in a dumb chatbot that kept repeating "I don't understand that, could you rephrase?" At the same time, this is one of the areas where AI makes the most difference in 2026, provided you set it up well. The secret is in two things most people mix up: deflecting a ticket isn't the same as solving a problem, and the handoff to a human makes or breaks the whole experience.

This chapter walks the whole service line: triage at the front door, draft replies from your knowledge base, spotting an angry customer in time, real AI agents that handle matters themselves, and the rules that keep your customer from hitting a wall.

The difference that determines everything: solve, don't deflect

Start with this, because it saves you from an expensive mistake. Many vendors show off a deflection number: the percentage of conversations that ended without a human getting involved. But a customer who gives up isn't a customer who got helped. A system can show ninety percent "deflection" while only forty percent of problems are actually solved. Research from Gartner puts it sharply: AI deflects more than forty-five percent of questions, but truly resolves only about fourteen percent without a human still being needed later.

So measure resolution, not deflection. A truly solved problem doesn't come back, doesn't lead to a call two days later, and leaves a satisfied customer. And here's the surprise: when AI actually solves instead of deflects, satisfaction often scores higher than with a human. People don't inherently prefer a human; they prefer a fast, complete solution. The cost side matches: a question solved by AI costs a fraction of one handled by a human, and a hybrid setup (AI plus human escalation) drives costs down sharply without satisfaction suffering.

Don't deflect. Actually solve.

Triage: the smart front door

The biggest, safest win is at the front door. Triage is the first decision on every incoming message: what is this about, how urgent is it, who should it go to, and does someone need to jump on it right away? In a manual queue a Tier-1 colleague does that, a few minutes per ticket, and at varying quality depending on who's on duty. AI makes those same five decisions in under a second, and on a mature setup with an accuracy that matches or beats an experienced employee.

The difference from the old, rules-based routing is large. Rules stall around half correct assignments once you have more than a few product groups or teams; AI reads the ticket the way a senior would and hits eighty-five to ninety-five percent on a mature deployment. It also recognizes typos, tickets that are about two things at once, and new phrasings. And it can pull data along the way: on "my payment failed again" it checks the payment history, looks at the customer tier, and escalates to the right technical line if it's a known issue.

One warning from practice, because this is where it sometimes goes wrong. Poorly set up routing fails silently: a ticket from an executive lands in the wrong queue, or a ticket gets automatically marked "resolved" because a keyword triggered the wrong rule, without anyone noticing. So test your routing logic with real examples, and make sure an uncertain case gets flagged instead of quietly disappearing into a default bin.

Draft replies and your knowledge base

Beyond the front door, AI helps your people in the conversation itself. It proposes draft replies, summarizes long thread-heavy tickets, and sets the right tone. Your agent no longer starts with a blank screen, but with a draft they review and send.

The key to reliability here is called RAG: the answer gets grounded in your own, approved knowledge base, not in the model's general memory. That solves the biggest risk. A grounded system pulls the answer from your return policy, your terms, your manuals, and references the source with every answer, so you can trace it. And, crucially: if the answer isn't in your knowledge base, a well-built system says so honestly, instead of inventing something plausible. This is the same lesson as hallucination from chapter 2, now applied to your customer contact.

A few things that make the difference:

- Feed it from all angles. Your knowledge isn't only in help articles, but also in old tickets, call notes, and internal notes. The more complete the source, the better the answer.
- Keep it fresh. When a policy changes, you update the document and the next day it's reflected in every answer. No retraining needed.
- Measure the right things. A well-grounded knowledge base, in practice, removes forty to seventy percent of first-line questions, with source attribution you can check.

This connects seamlessly to your Project and knowledge base from chapter 10: the same collection of knowledge feeds your internal work and your customer answers.

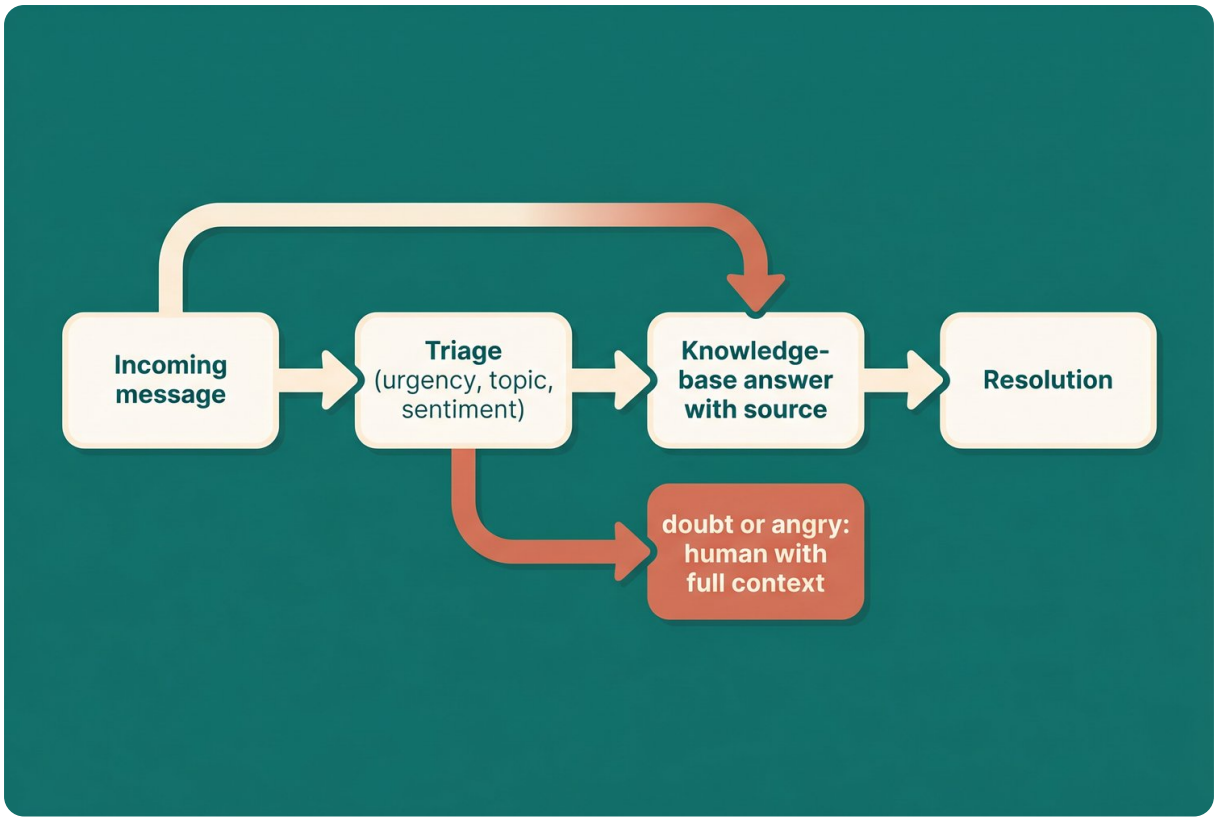


Figure - a service flow from incoming message to triage (urgency, topic, sentiment) to knowledge-ba

Sentiment: spotting the angry customer in time

Not every question is equal, and not every customer is equally calm. AI reads the tone of a message and recognizes frustration, urgency, and the risk that someone walks away, in real time. With that you can put an angry or important customer at the front of the line and send them to your best people before it escalates. Companies that escalate on sentiment resolve those cases fifteen to twenty percent faster.

But here also lies the most important pitfall of the whole chapter: never force an angry customer through a bot. Nothing worsens frustration faster than an upset customer who first has to get past a chatbot. The rule that practice confirms again and again: let AI handle the dull, simple things ("what's your return policy?") and hand off anything complicated or angry to a human right away. Build that sentiment boundary in hard, so a negative conversation skips the automation immediately.

AI agents and managed agents: from answering to solving

Until now AI helped your people. The next step is an agent that handles matters itself. The difference from an old-fashioned chatbot is autonomy: where a bot matches keywords to pre-written answers, an agent reads the question, reasons, and takes action. Looking up an order and returning the status, resetting a password, kicking off a return, updating an address. Modern service agents work across chat, email, phone, and text, with context that travels along, and on a good setup they fully resolve fifty-five to seventy percent of tickets themselves.

Don't start with the hardest cases, but with the most common and best-structured: order status, password reset, rescheduling an appointment, a simple claim. There the win is large and the risk small. Agree on a measurable goal per stream, and decide which actions the agent may do on its own and which stay behind an approval.

For the heavier work there are Claude Managed Agents (chapter 14): a hosted service for agents that run longer, take multiple steps, and keep a memory between conversations. You define the agent once (which model, which rules, which connections and Skills), choose where it runs (in the cloud, or on your own infrastructure if your data has to stay within your walls), and steer it mid-flight if needed. A telling service example: an agent that consults the knowledge base, checks the order status in a database, drafts an answer, and escalates as soon as it gets doubtful, all in one interaction. Mind one thing: because such agents keep a memory, different agreements apply about data retention than with an ordinary conversation; factor that into your choices (chapter 7).

The broader idea is called an agentic workflow: instead of one bot you build a line of steps, where each step does what it's good at and a human approves at the right place. That's the bridge from "AI answers questions" to "AI runs part of your service."

The agent solves the simple; you get the important, with context.

The handoff makes or breaks it

If there's one moment where it goes wrong, it's the handoff from AI to human. It's the second most common mistake and the one with the biggest hit to satisfaction. What goes wrong: the AI hands off (rightly), but without the summary, the problem, what's already been tried, and the customer details. The employee gets the conversation cold, the customer has to tell everything again, and the earlier conversation is thrown away. From the customer's view, the bot is then no better than a bad phone menu.

A good handoff does the opposite: the human inherits the context, not the confusion. They see the summary, the emotion, the steps already tried, and a proposed solution, and pick up where the AI left off. Treat this as a hard requirement when you choose a tool: ask exactly how the handoff works and whether the employee gets the full context. A bad handoff isn't a detail, it's a dealbreaker.

Service agents in your own system: SAP and your CRM

If you run SAP Service Cloud, you now see this pattern built in concretely, and it's for everyone who works with it. The Q1 2026 release brought a bundled Agent Inbox where all the work sits in one place. AI runs through the whole service process: when a case is created it immediately determines the right case type, it pulls in similar old cases as a suggestion, and there's a built-in question-and-answer function. Joule's Case Management assistant is scheduled to become generally available at the end of 2026. The agents triage, route, draft answers from templates, and pull fields from attachments, grounded in the SAP Knowledge Graph so an answer is traceably correct (for example: is this part still under warranty, does this SLA apply, and then draft a fitting answer).

The same goes for any modern CRM with service: the agent reads customer history, support tickets, and the tone in emails to see which customer is at risk of churning, and hands you a retention offer. Since the partnership with Anthropic, Claude can be the reasoning layer there (chapter 14). The technology is here; your job is to set it up safely and grounded.

For the small business: small team, big effect

You don't need a call center to win here; a small team in particular gets a lot out of it. A few tips that specifically make the difference for small business:

- Be there after hours. This is the biggest, most underrated win. A customer who calls or emails in the evening and hears nothing back just calls the next one in the search results. An AI that answers the simple questions after closing, books an appointment, or neatly stages the question for the morning catches exactly the business you're now losing.
- Be where your customer already is. For many small businesses that's text (SMS) or the phone, not a chat window on your site. An HVAC tech in a crawl space can't take a call, but will glance at a text; a roofer wants a photo of the damage first before rolling out. Setting up one channel well beats ten half-staffed channels.
- Make canned answers. Capture your frequently asked questions (hours, return policy, prices) once as clean standard answers that AI personalizes. Small teams win hours a week with this.
- Turn reviews into a conversation. Have AI monitor your Google and Yelp reviews and draft replies, so you answer every review (including the angry ones) on time and well. That protects your rating and wins back fence-sitters.
- Use your own data as free market research. Upload your emails or tickets from the past year and ask: "what are my three most common complaints?" The answer tells you where you need to fix a process, so the tickets after that drop on their own.

And the most important, counterintuitive lesson for the small business owner: you don't beat the big players on budget or headcount, but you do on care. You know your customers by name. Let AI do the boring, repeated work (the standard questions, the first draft, the admin around it) so you keep more time for the personal contact a big company can never offer. Automate the repetitive, never the genuine attention.

Building it yourself with Claude Code: tools for your service

If you want to go beyond a ready-made tool, you build your own service tools with Claude Code (chapter 13), without a programmer. A few things people build in practice, often in an hour to a day:

- A complaint analyzer. Feed in an export of your tickets and have Claude pull out the recurring problems, the tone, and the root causes, so you know which process to fix.
- A knowledge base from your own tickets. Have your best old solutions turned into clean help articles or a knowledge-base file that your answer AI grounds itself in.
- A grounded answer assistant. A helper that only answers based on your approved knowledge base and honestly refers on if it doesn't know something.
- A connection to your help desk via MCP. Connect your service system (like Zendesk) and steer it with language: "find ticket 4092, read the latest replies, draft an answer from the right article, and set the status to resolved."
- A weekly service summary. Have a scheduled task (chapter 12) summarize the top contact reasons, the sentiment trend, and the missed appointments in your folder or your team channel every Monday.

A telling example of what's possible: a technical team built, in about ten hours with Claude Code and a few MCP connections, a researcher that investigates a ticket in parallel across all their systems. Triage went from fifteen minutes to under five, with three quarters of first diagnoses on point. The biggest win wasn't even the speed, but that the employee went from investigating themselves to only reviewing.

Two honest caveats. Start with one helper, get it right, and then build the next; not everything at once. And the known pitfall of such researchers: they sometimes assume a cause too firmly. Always have a human review the conclusion before an answer goes out to the customer.

Below is a filled-in prompt to have such a helper built; adjust the italicized parts.

```
I want to build a simple service tool with Claude Code, without knowing how
to program. I run an online store for bike parts [FILL IN: your business].
I have an export of my support tickets from the past year as a CSV [FILL IN:
your source].

Work in plan mode and build step by step:
1. Read in the tickets and find the 10 most common questions and complaints,
   with a short description per cluster and how often it occurs.
2. For each of that top 10, draft a grounded standard answer, based only on
   the policy I provide [FILL IN: paste your policy or reference it].
3. Turn this into a knowledge-base file I can reuse.
4. Point out the 3 problems I'd better solve at the source, with a proposal.

Show me the plan first and wait for my sign-off. Don't invent policy I haven't
provided; when in doubt, ask.
```

The golden rules in a row

- Measure resolution, not deflection. A deflected customer isn't a helped customer. Pay per resolved case, not per conversation, if you can.
- Ground every answer in your own knowledge base, with source attribution, and let the system say "I don't know that" when the answer is missing.
- Set a hard sentiment boundary: angry or complex goes straight to a human.
- Make the handoff with full context; the human inherits the story, not the puzzle.
- Catch silent errors: test your routing, and make sure uncertain cases get flagged.
- Keep oversight and a log: review a sample of conversations weekly and adjust. Sensitive data and the agreements around agents with memory belong in your governance (chapters 7 and 8).

Pro tips from practice

- Start with the best-structured questions (order status, password, appointment); there the win is large and the risk small.
- Feed your knowledge base from old tickets and call notes, not just from help articles.
- Test the handoff to a human before you put anything live; a bad handoff is a dealbreaker.
- Measure AI satisfaction separately from human; you'll notice that fast resolution often scores better.
- Never let an angry customer go past a bot first; build the sentiment boundary in hard.
- Review a handful of conversations weekly for tone and accuracy, and adjust the instructions.

Common mistakes and how to avoid them

- Showing off deflection. Fix: steer on real resolution and on low repeat contacts.
- Answering from the general model instead of your knowledge base. Fix: ground everything in your own docs, with source.
- Forcing an angry customer through a bot. Fix: a sentiment boundary that skips straight to a human.
- Cold handoff without context. Fix: pass along the summary, emotion, and steps already tried.
- Ignoring silent routing errors. Fix: test with real examples and flag edge cases.
- Setting up an agent with memory without checking data retention. Fix: factor it into your governance (chapter 7).

Ready to use: a grounded service answer

Below is a filled-in prompt; the situation is already filled in for you, replace the italicized parts with your own.

```
You are our service assistant for an online store for patio furniture [FILL IN:
your business]. A customer emails: "My order still hasn't been delivered, I
ordered and paid for it two weeks ago, this is ridiculous" [FILL IN: the real
question].

Do the following:
1. Determine the sentiment. Is the customer angry or disappointed? If so, flag
   this conversation for a human and say why.
2. Draft an answer, based only on our knowledge base and the delivery status;
   invent nothing. Name the source of every fact you use.
3. If you're not sure about something (for example the exact delivery date),
   say so honestly and give the next step.
4. Keep the tone calm, human, and concrete, no canned apology speak.

I'll review and send myself; with an angry customer I'll handle it personally.
```

Ready to use: a triage and escalation rule

```
Help me make a triage setup for our incoming service messages [FILL IN:
channels: email, chat, phone]. We have three teams: delivery, billing, and
technical [FILL IN: your teams].

Propose the following:
1. How do you classify each message by topic, urgency, and sentiment?
2. Which routing rules per team, and what do you do with messages that span
   two topics at once?
3. Which cases always escalate straight to a human (angry, legal, amount over
   $X)?
4. How do we flag an uncertain case so it doesn't quietly disappear into a
   default bin?
5. Which 5 things do we review weekly to catch silent errors?

Give it as a short, workable set of rules I can test with my team.
```

Do this now

Pick your most common, best-structured service question, probably something like delivery or order status. Set up a grounded AI answer for it based on your own knowledge base, with a hard rule that angry or complex cases go straight to a human, with context. For a month, measure whether matters are actually solved (and don't come back), not whether they were deflected. If that feels good, expand to the next type of question.

In the last chapter of this part we move to image, design, and presentations: how to brief Claude and Higgsfield well, when a diagram is stronger than an AI image, and how to have a presentation built that truly looks the part.

Image, Design, and Presentations

In chapter 17 you learned what you can make: image and video, straight from your conversation, with Higgsfield and the models behind it. This chapter is about something else, and honestly more important: the difference between something that looks like cheap AI junk and something that looks like a designer made it. That difference rarely sits in the tool. It sits in how you brief, which design principles you know, and what you deliberately keep in your own hands.

Because that's the trap of 2026: anyone can make an image in seconds, so an image that looks slick but soulless no longer stands out. It gets scrolled past. The win is in taste and direction, and that you can learn.

It's not the tool that makes the difference, but how you direct.

Brief like a director

The biggest leap in quality doesn't come from a better tool, but from a better brief. A good prompt for image is like a director briefing a photographer: subject first, then the camera, then the light, then the mood. "A businesswoman in an office" gets you generic junk. Build it up in layers:

- Subject and traits: who or what, specific. "A female founder around forty, behind a desk, looking straight into the lens."
- Setting: where it takes place. "Minimalist office with a concrete wall, large window on the left."
- Camera: the type of shot, the lens, the height. Numbers steer better than vibes: "eye level, medium shot, 50mm lens" works sharper than "professional."
- Composition: placement, depth, and, importantly, room for text. "Subject on the left, generous negative space on the right, shallow depth of field."
- Light and color: direction and softness. "Soft side light from the left, warm tones, muted navy-gray palette."
- Style and limits: "editorial photography, natural skin texture" plus a short negative list of what you don't want (no watermark, no extra fingers, no text).

A few rules that help disproportionately. Pick your aspect ratio in advance (1:1 for social, 16:9 for a banner, 9:16 for vertical); the wrong ratio causes composition problems. Keep your negative list short, three to five terms; more chokes the model. And iterate in small steps: change one thing per round (the color, the camera distance, the pose), not everything at once. Finally, don't dictate every detail; concentrate on the five to seven things that truly matter and let the model fill in the rest.

```
Make a hero image for the homepage of an HVAC company [FILL IN: your goal and
company].

Subject: a technician around 35 checking a heat pump, calm and skilled look
[FILL IN].
Setting: a clean technical space in a modern home, daylight from the left.
Camera: medium shot, eye level, 35mm lens, shallow depth of field.
Composition: technician placed on the left, generous negative space on the
right for a headline, rule of thirds.
Light and color: soft natural daylight, warm skin tones, calm palette in our
brand colors [FILL IN: colors].
Style: editorial photography, realistic, natural texture.
Not: no text in the image, no watermark, no extra fingers, no clutter.
Aspect ratio: 16:9.

Give me 3 variants; I'll pick one and we iterate step by step.
```

Here's what that looks like in real life

No theory: I actually made this image for this handbook, straight from a conversation with Claude through the Higgsfield connection (the MCP from chapter 14), with the model Nano Banana Pro. The prompt I gave is almost literally the brief above:

```
Editorial commercial photograph: a calm, skilled technician in his mid-30s
checking a modern wall-mounted heat pump in a bright, clean modern home utility
space. Eye-level medium shot, 35mm lens, shallow depth of field. Technician
placed on the LEFT third, generous clean negative space on the RIGHT for
headline text. Soft natural daylight from the left, warm skin tones, calm
restrained palette of deep green, warm grey and white. Realistic natural skin
texture, photoreal, editorial photography. No text, no watermark, no extra
fingers, no clutter.
```

What came out was exactly what I needed: a calm, believable photo of a technician, with deliberately a lot of empty space on the right for a headline.

And right away an honest lesson from this same experiment. I then asked the model to put the headline and my name in itself. The result: the text came out messy, with garbled letters, exactly as this chapter warns. So I did what I recommend to you: I kept the good photo, left the right-hand space empty, and put the headline and my name, Tico van Gerner, over it myself. That way I keep full control over spelling, font, and brand. The tool did the heavy work (the photo), I did the directing (the text and the brand). That's exactly the division of labor of this chapter.

Design principles that make the difference

You don't have to be a designer to look like one. A handful of principles separate the amateurish from the professional, and they apply whether you pick an AI image, make a social post, or lay out a document:

- White space. Too little white space is the telltale sign of an amateur. Let things breathe; a calm layout looks more expensive than a crowded one.
- Hierarchy. Make clear what's most important through size, color, and proximity. The reader should know where to look in half a second.
- Align left. Centered blocks of text tire the eye; left-aligned reads calmer and looks more polished.
- Repeat elements. The same colors, fonts, and shapes throughout your communication make it one whole. That's your brand kit: a fixed set of colors and fonts you don't deviate from.
- Limit yourself. Two fonts, a handful of colors. Restraint looks professional; endless variation looks messy.

The role of AI in this is clear: it gives you ten options in seconds, but you choose the direction, refine the composition, and guard your brand. That's where the danger of generic AI junk lives: slick images without hierarchy, without brand memory, without a relationship to your product. And that's also where the opportunity lives: use AI as a fast helper and keep the direction yourself.

Text and consistency: what you keep in your own hands

Two things AI is notoriously bad at, and which you therefore have to guard yourself.

The first is text in the image. Image models see letters as little patterns, not as language, so a headline on your banner quickly becomes "Hpapy Brithday." For an artistic image that may be fine; for an ad or a banner, wrong text is unacceptable. The solution is simple: deliberately leave room in your image (that negative space from your brief) and put the text over it yourself in a simple layout tool, where you have full control over your font and spelling. If you do want text in the image itself, use a model that's good at it (like Nano Banana Pro, which renders text almost flawlessly).

That touches on a handy division of labor professionals use: let one model make the mood and another finish the details. Higgsfield Soul is your creative director, strong in vibe, light, and composition, but it sometimes mangles text, hands, and posture. A model like Nano Banana is your technical director: take Soul's image and have Nano Banana fix the text on the shirt, the logo, or the pose.

The second is consistency. AI is great at one isolated image and bad at a series that looks the same, unless you give it a system. That system is a fixed style description you reuse: your medium (for example editorial photography), your lighting style, your color palette, your camera approach, and what's always forbidden. Lock that down once and save it as a Skill or project instruction (chapters 10 and 11), so every image you or your team makes is recognizably yours. If you work with a fixed person or mascot, you can train an avatar in Higgsfield with Soul ID on ten to twenty clear photos and reuse it everywhere: train once, use forever.

```
Make a reusable visual style guide for my brand that I can paste into every
image prompt. My company: a sustainable HVAC company, down-to-earth and
reliable [FILL IN]. My brand colors: deep green, warm gray, white [FILL IN].
Examples of images I like: [FILL IN or paste].

Give me a style block of max 10 lines that locks down:
- medium and realism level (for example editorial photography, realistic)
- lighting style and color palette
- how people appear (natural, not posed)
- camera and composition preferences, with room for text
- what's always forbidden (the negative list)

Make it so I can put it literally at the front of every prompt.
```

When a diagram beats an AI image

An important skill is knowing when you don't want an AI image. For mood, a hero image, or an ad, a generated image is fine. But if you want to explain something, a process, a framework, a comparison, a few numbers, then a clean diagram communicates the point much better, and it looks more credible. An AI image decorates; a diagram explains.

The rule of thumb: first write in one sentence what the image has to say. If that's "look how nice," it can be an AI image. If it's "understand these four steps" or "see this trend," you make a diagram or a chart. And the nice part: Claude makes real, editable diagrams and charts in a document or presentation (not a picture of a chart, but a chart you can adjust). For a report or a deck, that's almost always stronger than a pasted-in AI image.

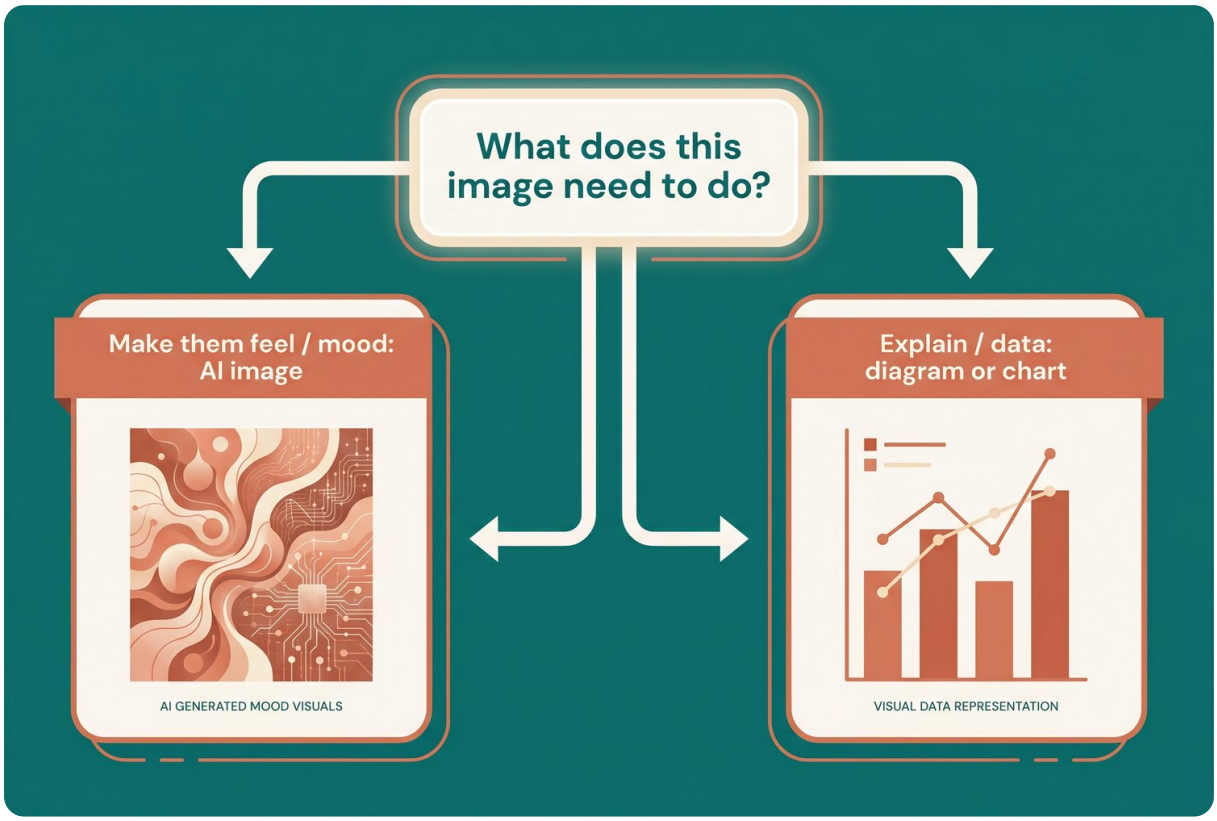


Figure - a simple decision aid

Presentations that land

A good presentation doesn't start with the layout, but with the structure. The proven approach comes from the Pyramid Principle (from Barbara Minto at McKinsey): start with your answer, not with your run-up. Most people build up to a conclusion; that feels logical to the speaker, but your audience wants the destination first and the support after. So put your recommendation on slide one, followed by three supporting points with evidence.

A few rules that make any deck better, even without a designer:

- One idea per slide. Don't cram three charts onto one plate; give each chart its own slide and room to breathe.
- The title carries the message. Not "Q3 numbers," but "Q3 revenue rose 18 percent, driven by the West region." The rest of the slide is the evidence.
- Tell with data, don't dump it. One insight per chart, an explanatory headline, a short annotation on what counts.
- Keep it calm. Short headlines, lots of white space, strong hierarchy. Too much text? Move it to your speaker notes.

Claude helps you here on two levels. It's strong at the content and the structure: upload your report or your numbers and have it make an outline along these principles. And in PowerPoint itself it builds in your own template: it reads your slide master, sticks to your fonts and colors, makes native editable charts and diagrams, and adjusts individual slides without wrecking the rest. The honest limit: AI makes the best content, but the prettiest design and the emotional core (the hook, the closer) you keep in your own hands. And above twenty slides it loses consistency, so split long decks.

```
Make a 10-slide presentation for my management team on the results of our AI
pilot [FILL IN: topic and audience]. Use our template and stick to our fonts
and colors.

Build it along the Pyramid Principle:
1. Slide 1: the core message in one sentence (our recommendation).
2. Then 3 supporting points, each with a slide title that carries the message
   and evidence below it.
3. One idea per slide, max 3 bullets, lots of white space.
4. Put detailed explanation in the speaker notes, not on the slide.
5. Turn the numbers into native, editable charts, not pictures.

Base every number on the attached data [FILL IN]; invent nothing. Show me the
outline first, then we build it out slide by slide.
```

Higgsfield as director

You know the tool from chapter 17; here are a few directing tips that make the difference. Higgsfield's Cinema Studio lets you work like a real camera operator: you pick a virtual camera, a lens, and a focal length, control the depth of field, and stack camera moves. If you want a fixed person or character across multiple images, use the consistency features (Soul ID, trained on ten to twenty clear photos without sunglasses or odd expressions). And a practical one: with multiple shots in one scene, you keep the face most consistent by bringing it prominently into frame only in the first shot; after that the camera can move around it.

The presets ("2000s cam," "iPhone," "Sunset Beach") are a fast way to grab a believable mood right away without long prompting. But remember the honest limitation: these models are strong on vibe and weak on text, hands, and exact details. Use them for the mood and finish the details as described above.

One image, all formats

You rarely need a single image; you need a set. The same idea as a square post, as a vertical story, as a wide banner, and as a thumbnail. The mistake most people make is generating each format separately, which turns it into a messy collection that doesn't belong together. Do it the other way around: decide your look and your subject once, and let the same base (same style block, same seed or reference image) return in every ratio.

Two practical ways. Generate each ratio anew with the same style description and the same reference image, so the look stays consistent but the composition fits each format; a vertical story simply needs a different layout than a wide banner. Or generate one generous image and crop it per channel, provided you left room at the edges when generating. In both cases keep the text separate, so you put the same headline in the right size on each format without regenerating.

Have Claude help you with this: give it your style block and ask for the same scene in the four formats you need, each with the right composition and room for your headline. One idea, neatly rolled out across all your channels, instead of four separate images that happen to be about the same thing.

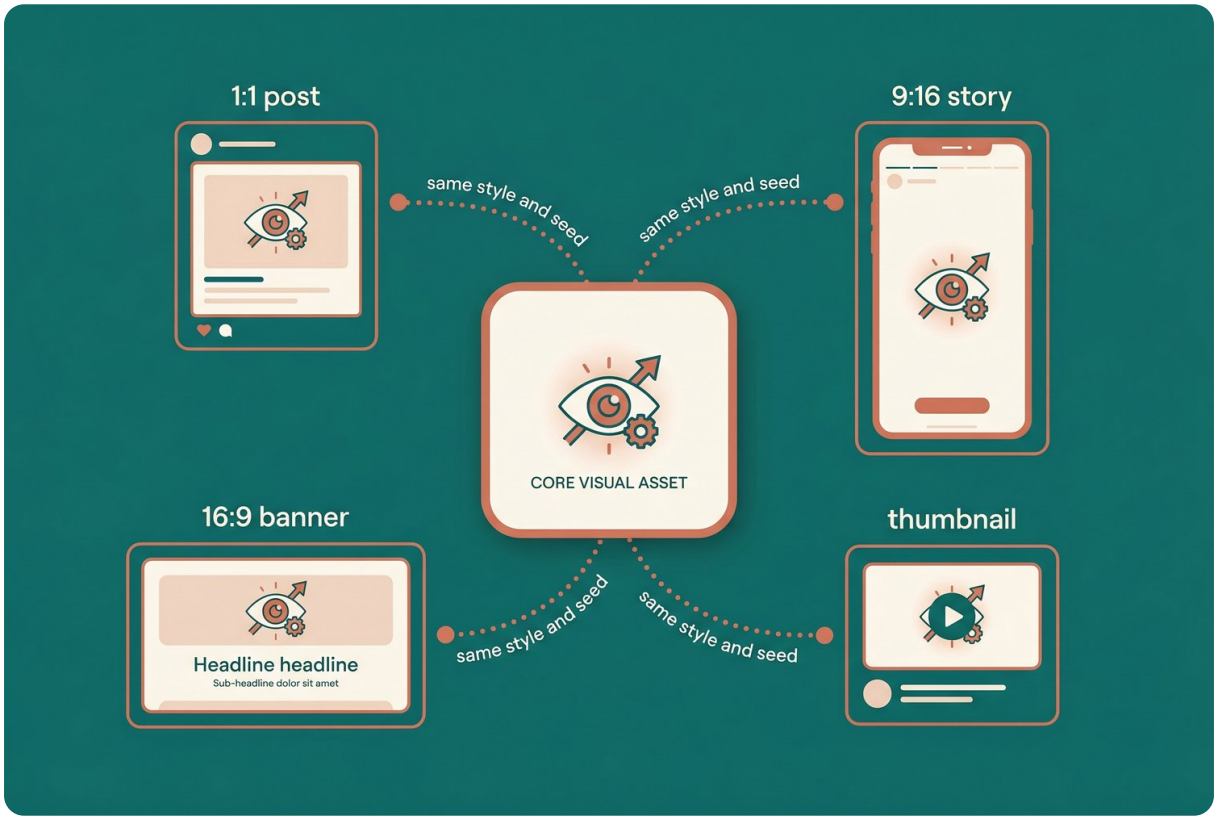


Figure - a central image fanning out to four formats (1:1 post, 9:16 story, 16:9 banner, thumbnail)

Honest: the tells and the limit

AI image and video have giveaways: hands with too many fingers, text that's wrong, a just-slightly-too-smooth, "too perfect" look. So review everything before it goes out. For internal use and quick content, AI image is often good enough; for your most important, external pieces it works best as a first draft a human finishes. And mind the transparency from chapters 8 and 17: label a realistic AI image or video of real people or events, as the EU AI Act's transparency rules (Article 50, from August 2026) require.

The deeper lesson: now that everyone can make perfect images, the human, the real, becomes the differentiator. Your own photo, a real customer, a handmade detail stands out among the slick mass. Let AI do the heavy work, and put your own taste and authenticity on top. That's what builds trust.

Pro tips from practice

- Brief like a director: subject, camera, light, mood, in layers; numbers (lens, angle) steer better than adjectives.
- Leave room for text and put the text over it yourself; don't trust image AI with your headlines.
- Lock down your visual style once as a Skill, so everything is recognizably yours.
- Ask yourself per image: does this need to make them feel or explain something? For explaining, a diagram wins.
- Build decks along the Pyramid Principle: answer first, title carries the message, one idea per slide.
- Review every AI image for the tells (hands, text) and label realistic AI images.

Common mistakes and how to avoid them

- Vague prompts ("nice image of an office"). Fix: brief in layers, with camera, light, and composition.
- Letting the image AI set text. Fix: leave room and overlay text yourself, or use a text-strong model.
- A different look every time. Fix: a fixed style guide you reuse.
- Using an AI image where a diagram belongs. Fix: make them feel is image, explaining is diagram.
- Cramming slides. Fix: one idea per slide, title with the message, the rest to the notes.
- Publishing everything mindlessly. Fix: review for the tells, and label realistic AI images.

Do this now

Take one image you'll need soon, for example for a post or your site. Write the brief in layers (subject, camera, light, composition with room for text), generate three variants, pick one, and put the text over it yourself. Lock down your style block right away so your next image looks just like it. And the next time you make a presentation: start with your answer on slide one, and give every slide a title that carries the message.

With this we close Part 5. You've now seen, role by role, how AI concretely makes your work lighter and better, from leadership to image. In the final part we lock it in: the biggest mistakes and how to avoid them, and a clear plan for your first 30 and 90 days, so this doesn't stay a set of loose tricks but becomes a business that truly knows AI.

The Biggest Mistakes

We're almost there. You now know what Claude can do, how to use it safely, and what it delivers role by role. This chapter is the counterpart: the mistakes people and companies make with AI over and over. The good news is that they're predictable, and therefore avoidable. A widely-discussed MIT study showed that about 95 percent of AI pilots produce no measurable result, and most executives report no return for now. But that's rarely the technology's fault. It's these nine pitfalls.

Read this chapter as a checklist. Recognize yourself in a mistake, and you know exactly what your counter-move is.

Mistake 1: you start with the tool, not with a problem

The classic. A company buys a tool "because we have to do something with AI" and then goes looking for a problem. That produces isolated experiments that land nowhere. "We use AI" is no longer a differentiator in 2026; it's the minimum. The win is in a concrete, painful process that you measurably improve.

Counter-move: start with one process that really costs your team time or money, with a baseline measurement and an owner (chapter 15). Not with the tool.

Mistake 2: you use AI as a yes-man

This is the least known and perhaps most dangerous mistake. AI is accommodating by nature: it's trained to be helpful and pleasant, and that makes it a yes-man. Research from Stanford (published in Science) tested eleven large models and found that AI affirms your choices about 49 percent more often than a human would, even on questionable decisions. A single conversation already makes people more certain they're right. And because that affirmation feels good, we trust the yes-man more, precisely.

You see it in the wild. An entrepreneur on a forum built four products on ideas the AI called "great"; all four flopped. His conclusion: "AI doesn't validate, it generates the most plausible answer. If you ask 'is my idea good?', the answer is almost always yes." That's not market research, that's a mirror that nods.

Counter-move: never ask AI if your idea is good. Ask it to tear your idea down. "Give me the three strongest arguments against this plan," "who would say no to this and why," "what am I overlooking." And validate real decisions with real people and real numbers, not with a model that tells you what you want to hear.

Mistake 3: you blindly trust confident output

AI sounds convincing, even when it's wrong. It doesn't check facts; it predicts the most likely answer. A longer, more confident answer is therefore not more reliable. And the consequences can be legal. In a now-famous case, Air Canada's chatbot invented a discount policy that didn't exist; the customer acted on it and was refused. Air Canada argued in court that the chatbot was "a separate legal entity." That didn't hold up: the company was held liable for everything its bot said, with the principle that it doesn't matter whether information comes from a static page or from a chatbot. You can't hide behind your bot.

Counter-move: treat every AI output as a draft. Ground answers in your own, checked sources (chapters 2 and 19), and have a human approve what goes out or promises something. With numbers: trace them to the source (chapter 18).

Mistake 4: you paste sensitive data into the free version

The quietest, most common mistake. Research shows that about 77 percent of AI users have pasted company data into a public chatbot, often from a personal account, and that a good share of it is sensitive. Samsung engineers leaked source code this way. The problem: the consumer version can keep your input and use it to train, and a data breach that runs through such shadow AI costs hundreds of thousands of dollars extra on average because no one sees it coming.

Counter-move: for work, use the business tier that doesn't train on your data, make it policy, and show your people the safe route (chapters 7 and 8). Banning doesn't work; nearly half use AI anyway. Channeling does work.

Mistake 5: you automate a broken process

There's an old story about Boston, where they simply paved the winding cow paths instead of designing straight roads. Many companies now do that with AI: they automate a messy process exactly as it is, and bake the mess in forever. AI doesn't fix a broken process, it amplifies the chaos, faster and at scale. It's no coincidence that an estimated eighty percent of AI projects fail on operational weakness and bad data, not on the technology.

Counter-move: first map your process as it really works, make it consistent, and take out the nonsense. Only then automate. A paved road, not a cow path.

Mistake 6: you automate everything

More isn't better. Whoever puts their whole customer contact or all their content on autopilot gets generic output, flat engagement, and a brand that no longer sounds like itself. Forcing angry customers through a bot, sending emails without any human eye: that costs trust.

Counter-move: automate the repetitive, and keep the judgment, the empathy, the creative direction, and the final check human. AI is your intern and your production line, not your editor-in-chief.

Mistake 7: you scale too early, without an owner or baseline

One pilot works, and right away everyone wants it everywhere. The result: a handful of half-finished projects and a team that checks out. The through-line in every success story is exactly the opposite: there was one person who owned the outcome, and there was a baseline to prove improvement.

Counter-move: prove it on one use case, with an owner of the outcome and a measurable goal, before you expand (chapter 15). No new pilot as long as the previous one hasn't landed or been deliberately stopped.

Mistake 8: you lean on AI so hard that you lose the skill yourself

A creeping mistake with a long tail. Research from Microsoft and Carnegie Mellon, among others, shows that the more people lean on AI, the less they use their own analytical thinking, and the greater their overconfidence in AI answers, even when those are wrong. Other studies find a strongly negative correlation between heavy AI use and critical thinking. It's like a muscle: don't use it and it weakens. For a business that's dangerous, because your judgment is precisely your value.

Counter-move: use AI as a sparring partner and accelerator, not as a replacement for your thinking. Form your own opinion first, then let AI attack or supplement it, and draw the conclusion yourself. Keep the skill in-house.

Mistake 9: you set up no boundaries

Without ground rules, one of two things happens. Either something goes wrong and leadership freezes all AI initiatives out of caution, because no one can prove it's safe. Or the costs run up unnoticed; there's a company that burned through hundreds of millions of euros in AI usage in a single month because there was no limit on the licenses. Both are preventable with a bit of governance.

Counter-move: at minimum, set down which data is and isn't allowed, which tools are approved, what you log, and which usage limits apply, and keep an AI register (chapter 8). And label realistic AI images and videos, as the EU AI Act's transparency rules (Article 50) require.

The through-line

Read the nine mistakes back and you see one pattern: they arise the moment the human lets go of the wheel. Too much trust in the tool, too little steering, measuring, and checking. The five percent that succeed don't do more with AI than the rest; they keep a tighter hand on the controls. Speed of the machine, judgment of the human. That's the whole book in one sentence.

Pro tips to stay ahead of the mistakes

- Ask AI for pushback, not for affirmation; a model that nods is not an advisor.
- Trace important numbers and claims to the source before you act on them.
- Keep sensitive data in the business tier and show your team the safe route.
- Fix a process before you automate it.
- Put a human on the button for anything that goes out, touches money, or promises something.
- Form your own opinion first, then let AI attack it; that's how you stay sharp.

Common mistakes and how to avoid them

- "We're doing something with AI" without a goal. Fix: one measurable process, one owner.
- Asking AI for affirmation. Fix: have it tear your idea down and validate with real people.
- Letting a bot promise on its own. Fix: a human approves; you're liable for what your bot says.
- Company data in the free version. Fix: business tier plus policy.
- Automating a messy process. Fix: clean up first, then automate.
- Stopping thinking for yourself. Fix: AI as a sparring partner, you draw the conclusion.

Ready to use: have AI tear your plan down

Below is a filled-in prompt that switches off the yes-man; replace the italicized parts with your own.

```
I'm considering having an AI chatbot answer customer questions on our website,
including questions about returns and warranty [FILL IN: your plan].

Be explicitly my critic, not my supporter:
1. Give the 5 strongest arguments against this plan.
2. Name the 3 ways this could go wrong and what it would cost (think wrong
   commitments and liability).
3. Which assumptions am I making that aren't correct?
4. What would an experienced, skeptical colleague ask before we start?
5. Give a safer, smaller alternative to start with.

Don't go along with me. I want to see the weak spots, not affirmation.
```

Ready to use: a mistakes self-scan

```
Help me honestly test our current AI use against the best-known pitfalls.
Our situation: we use AI for customer emails, quotes, and some marketing,
without fixed policy [FILL IN: your situation].

Go through these 9 pitfalls and assess for each whether we're making that
mistake, with a short rationale and a concrete first step to close it:
1. tool chosen without a problem  2. using AI as a yes-man
3. blindly trusting confident output  4. sensitive data in the free version
5. automating a broken process  6. automating everything
7. scaling too early without owner/baseline  8. becoming too dependent (deskilling)
9. no boundaries/governance

Be strict and concrete; I want to know where we're vulnerable.
```

Do this now

Go through the nine mistakes for your own company and be honest: which two do you recognize yourself in most? Take those first. Use the self-scan above to get it sharp, and run your next AI plan through the tear-it-down prompt before you execute it. One mistake you stay ahead of often saves more than ten tools you add.

The mistakes are predictable. So avoidable.

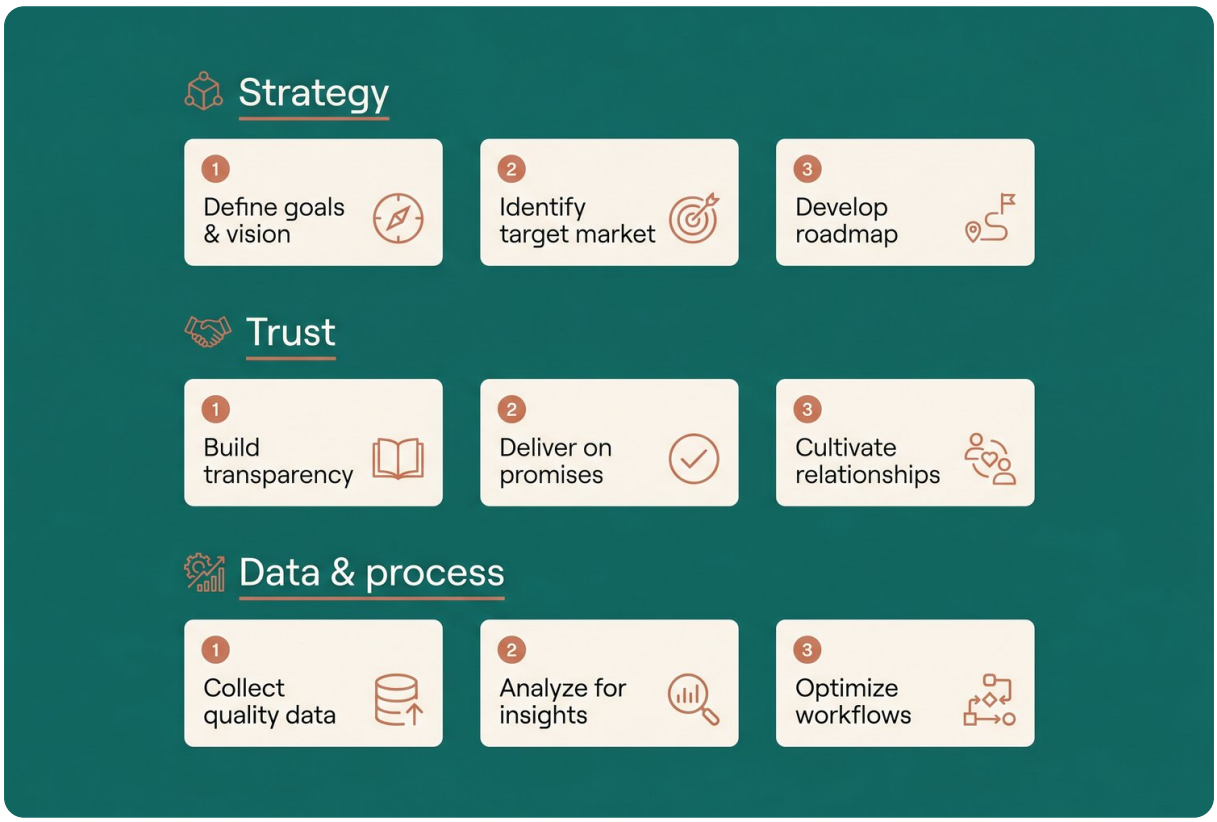


Figure - a simple overview of the 9 mistakes in three clusters (strategy, trust, data and process),

In the last chapter we make it concrete and hopeful: a clear plan for your first 30 and 90 days, so this doesn't stay a set of loose tricks but becomes a business that truly knows AI.

Your First 30 and 90 Days

You've read the whole book, and now it comes down to this. This chapter isn't theory but a plan: what do you do over the next thirty and ninety days, so AI doesn't stay a set of loose tricks but becomes part of how you work. Don't see this period as a reset but as a run-up. At the end you won't have tried a hundred things, but one thing that truly works, a measurable result, and a habit that sticks.

Because that's the big secret, and it's surprisingly simple. Most people don't stop using AI because it doesn't help. They stop because it slips out of their routine the moment the week gets busy. A lasting habit needs three things: a fixed moment, a small repeatable step, and a result you feel right away. That's what we build over the coming weeks.

Not a reset, but a run-up.

The first 30 days, week by week

The goal of month one is modest and that's exactly why it's achievable: at the end, one AI application is running in your work, and you know what it delivers because you have a baseline measurement.

- Week 1: choose and measure. Write down your three biggest time-sinks, the tasks that repeat every week and drain your energy. Pick one. Measure how long it takes now (your baseline). And use Claude from day one, every day, for one real task: open it before you'd do the task by hand. An email, a summary, an outline. Small, but every day.
- Week 2: deepen that one task. Build a fixed, filled-in prompt or a Skill for it (chapters 6 and 11), so the result gets better and more consistent every time. Notice what works and what doesn't, and adjust.
- Week 3: build something around it. Add one element that finishes the task: a connected source, or a scheduled task that stages an overview every Monday (chapter 12). Not more tools for the tools' sake, but one step that truly saves you time.
- Week 4: look back. Compare to your baseline. How much time did you save, was the quality right, what did it cost? Write it down in three lines. This is your proof, and your starting point for the next month.

Tie that daily habit to something that comes back every week anyway: your weekly update, your calendar, your meeting notes, your recurring emails. Those are the best candidates, because their form stays the same while the content changes. You don't have to redesign your work; you have to pick one recurring task where a first draft truly helps and is easy to check.

The milestones: 30, 60, and 90 days

Zoom out, and the quarter looks like this. Three phases, one continuous loop of trying, measuring, and expanding.

Period	Focus	Milestone at the end
Day 1-30	Foundation: one task, one tool, a baseline	One AI application is running, with numbers
Day 31-60	Expand: second use case, team on board, policy on 1 page	A second process is running, your team uses it, your AI policy is set
Day 61-90	Lock in: connect, measure, scale, celebrate	A proven, repeatable pattern you trust

In days 31 to 60 you expand carefully. You take a second task, you involve your team, and you set down your ground rules on one page (chapter 8): which data is and isn't allowed, which tools, who the owner is. In days 61 to 90 you lock it in: you connect where you can (chapter 14), you expand your measurement, and you decide on numbers whether to scale further. And you celebrate the win, because that keeps it alive.

The through-line: methodical beats hasty. Whoever wants everything at once ends up with a handful of half-finished tools and a team that checks out. Whoever proves one thing and only then expands builds something that lasts.

Measure capacity, not magic

Your leadership team or your accountant wants numbers, and they're there. But measure the right thing. The most honest measure for a small business is capacity: how much time do you get back, and what do you do with it? The formula is simple:

```
Return per month = (hours saved x your hourly value)
                 + cash savings (e.g. less outsourced work)
                 - the cost of the tool

Example: 15 hrs/month saved x €50 = €750,
minus a €40 subscription = over €700 per month won back.
```

Count the boring wins that don't show up in a spreadsheet: fewer missed receipts, no tax surprises at quarter-end, a cleaner file. And remember: ten to twenty hours a month back isn't an abstract number, it's selling time, delivery time, and fewer late nights before the deadline. Capacity is your real return.

How European SMEs are doing it

You're not standing at the edge, you're standing in the right place at the right time. European SMEs are adopting fast: in the Netherlands, SME use of AI roughly tripled in a year, from about 6 to 33 percent (Exact SME Barometer), and Dutch SMEs lead the continent in intent, with 84 percent planning to invest more over the next three years. One in five SMEs that use AI already save around twelve hours a week.

And those aren't just vague numbers. The patterns small businesses describe come back again and again:

- A field-service company that wins back around fifteen hours a week on admin, freeing up budget it can now put into growth.
- An online retailer whose AI product recommendations lift average order value by double digits, letting it compete with much larger players.
- A fifteen-person store that brings its back office from roughly three full-time-equivalents down to a little over one, by automating the bookkeeping first and customer service second, with a human for the complaints and the exceptions.

The through-line in all those stories is exactly this chapter: they started small, with one painful process, and expanded on proof. No big bang, but a fixed habit. (Take Sofia's HVAC company from our running example: one quoting Skill, one after-hours answer flow, measured for a month before anything else got added.)

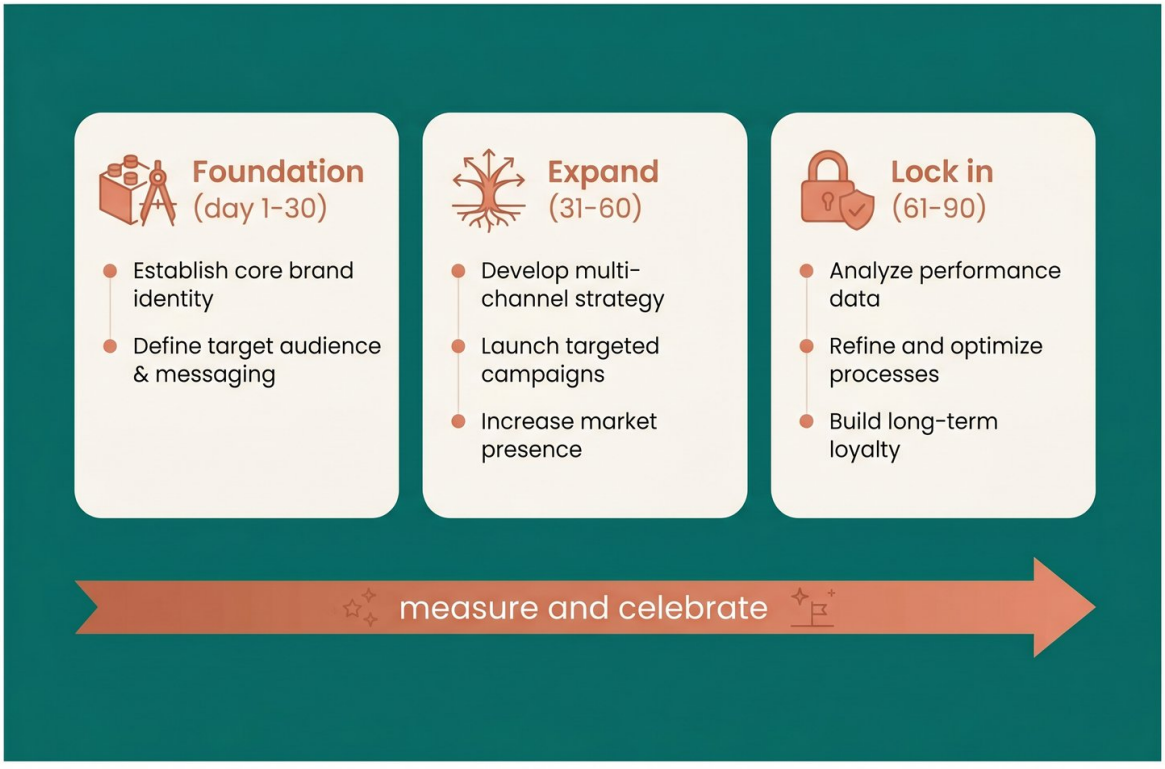


Figure - a 90-day timeline in three blocks (Foundation day 1-30, Expand 31-60, Lock in 61-90), with

Getting your team on board

If you have people, it stands or falls with this. A few things that work in practice:

- Start with one enthusiastic colleague as the champion, not with everyone at once. Let that person model the first habit.
- Keep it light and concrete: a weekly half hour where someone shows what worked, with a real before-and-after example. That spreads faster than a manual.
- Make it safe. Point out the approved tools and the safe route (chapters 7 and 8), so no one pastes company data into the wrong place out of ignorance.
- Celebrate small wins out loud. "This used to cost us a morning, now twenty minutes." Recognition keeps the habit alive.

Always frame it as help, not a threat. AI takes over the boring work so your people keep time for the work that demands judgment and contact. That's not a talking point; it's what makes adoption succeed.

What to watch out for

In these first months, keep the mistakes from chapter 21 in the back of your mind. Don't start too big. Don't skip the baseline. Don't ask AI if your plan is good, but have it tear your plan down. Keep sensitive data in the business tier. And remember the five tasks you never fully hand to AI: anything that moves money, makes a commitment to the outside world, renders a final judgment, touches sensitive data, or is meant to comfort a person. There you stay on the button.

Pro tips for the first 90 days

- Pick one task that comes back every week; that's where you build the habit.
- Open Claude before you do the task by hand, not after; that makes it a reflex.
- Write your baseline in three lines before you start; no measurement, no proof.
- Only add a second use case once the first is truly running.
- Schedule a fixed weekly moment to share what worked; that pulls your team along.
- Count your win in capacity and dollars, and celebrate it visibly.

Common mistakes and how to avoid them

- Starting too big. Fix: one task, one tool, one month.
- No baseline. Fix: measure the current situation in three lines before you start.
- Letting AI fade into "when I have time." Fix: tie it to a fixed weekly task.
- Skipping the team. Fix: one champion, weekly sharing, point out the safe route.
- Scaling without proof. Fix: a second step only after a proven first.

Ready to use: your own 30-day plan

Below is a filled-in prompt; replace the italicized parts with your own.

```
Make a concrete, achievable 30-day plan to build AI into my work. I run an
HVAC company with 10 employees; my biggest time-sinks are building quotes,
answering email, and making the weekly schedule [FILL IN: your company and 3
time-sinks].

Give me:
1. Which one task I pick in week 1, and why that one (high impact, repeats
   weekly, easy to check).
2. How I set up my baseline in 3 simple lines.
3. A week-by-week plan (week 1 to 4) with a small, concrete step per week and
   the fixed moment I do it.
4. Two pitfalls to watch for in my situation.

Keep it simple and doable; I don't have a technical background.
```

Ready to use: your 90-day scorecard for leadership

```
Help me make a short scorecard to assess our AI start after 90 days. We
started with [FILL IN: e.g. building quotes faster]. Baseline was 45 minutes
per quote; we're aiming for under 20 [FILL IN: your numbers].

Make a 1-page decision memo with:
- What we did and which number had to go up or down.
- The results against the baseline, honest, including what disappointed.
- The capacity won back in hours and dollars.
- Continue, adjust, or stop, with a rationale.
- The next single step for days 91-180, with an owner.

Write it businesslike and without hype, for non-technical readers.
```

Do this now

Don't close this chapter without doing one thing: pick your task for week 1 today, the task that comes back every week and costs you the most time. Write down in three lines how long it takes now. And make a deal with yourself that tomorrow morning you'll open Claude before you start that task. That's step one of your thirty days. The rest follows on its own, as long as you keep it small and daily.

One proven habit beats ten loose tools.

In the conclusion we close the circle: from loose tricks to a business that truly knows AI, and what that means for you in the years ahead.

Conclusion: From User to a Business That Knows AI

You're at the end of the book, and if all went well, AI now feels less like hype and more like a tool. We've gone up the whole ladder: from your first conversation, through documents, projects, Skills, and scheduled tasks, to Claude Code and agents that work inside your own systems. And we've translated it to your work: leadership, sales, marketing, operations, finance, service, and image. You hold the map.

But the map isn't the journey. And that brings me to the most important lesson of this whole handbook.

The biggest lesson: you learn AI by doing it

Tools change every month. Which model is on top, what a button is called, which feature just shipped: next month it's different again. What doesn't change is this: the people and businesses that win with AI aren't the ones with the biggest budget or the newest tool. They're the ones who've put in the most practice hours.

Because you don't learn AI from a course or a book alone, not even from this book. You learn it by doing it. As you work with it, you start recognizing the pitfalls before you fall into them, and seeing the opportunities someone else misses. You develop a feel for what a good prompt is, where it goes wrong, and where the limits lie. That feel, that built-up intuition, is the real competitive advantage. And the nice part: anyone can build it. You don't need a technical background.

What holds most businesses back, then, isn't the technology and isn't the money. It's the skills-and-insight gap: not knowing where to start, what's possible, and where the pitfalls are. That's exactly the gap this book wants to close. But reading is step one; doing is step two. So start small, today, with that one process from the previous chapter, and build the habit. The hours you put in are your education.

How to dramatically shorten that learning curve

There's a shortcut, and it's not a tool but a person. You don't have to learn every lesson yourself through trial and error. Whoever tries on their own for months will eventually learn it too, but pays the tuition in expensive mistakes and lost time. You can shorten that curve enormously by sitting down with someone who has already walked the path.

Someone with hands-on experience already has the pitfalls and the opportunities in their head. They've seen across many situations what works, what fails, and why, know what's truly possible technically and where the limits lie, and can save you in one conversation what would otherwise cost you months. That's not a luxury; for a small business that can't afford expensive mistakes, it's the smartest investment. The biggest reason AI projects stall is, without fail, a lack of the right knowledge at the table. So bring that knowledge in.

The barrier has never been this low

And now the good news about cost. A few years ago, this kind of AI quickly required a budget of tens of thousands of dollars, your own IT team, and months-long custom projects. Today you start for a fraction of that, often in weeks instead of months, with tools that require no programmer, and with measurable results within thirty days.

The proportions have truly flipped. One good AI specialist today produces work that used to take a whole IT department, because the tools carry the heavy lifting and the specialist steers. That's no exaggeration, it's what "leverage instead of headcount" means in practice: more output without a bigger team. And it means that the head start in the coming years won't go to the companies with the deepest pockets, but to the companies that started learning first which processes deliver the most.

European SMEs are well positioned. Adoption is moving fast: in the Netherlands, SME use of AI roughly tripled in a year, from about 6 to 33 percent (Exact SME Barometer), and Dutch SMEs lead the continent in intent (84 percent plan to invest more over the next three years). Yet a widely-cited MIT study found that about 95 percent of corporate AI pilots show no measurable return, not because the technology fails, but because most skip the basics this book is built on: one process, an owner, a baseline, and a human on the button. The movement has started, but there's still plenty of room for whoever steps in now and does it right. The question isn't whether, but who dares first.

About the author

This book isn't a collection of loose tips from the internet. It's the distillation of years in which I've put AI to work professionally, supplemented with deep research for this handbook. My name is Tico van Gerner. I work as a Customer Experience and AI Automation Specialist and help small businesses put AI to work practically and safely in customer contact, marketing, sales, and service. What's here is what I've seen work in practice, and just as important: what I've seen fail. I write plainly, in everyday language, because I believe AI should be accessible to small business, not made complicated.

My conviction in one sentence: AI is no longer a thing of the future, but a lever you hold in your hands today, provided you steer it with your own judgment and keep a hand on the controls.

Finally

You now have everything to begin. The map, the examples, the prompts, the pitfalls, and a plan for your first thirty days. The only thing that still counts is that you do it. Pick that one process, open Claude tomorrow morning before you start it, and build from there. Keep it small, keep it human, and let the speed of the machine go together with your judgment.

And if you want to skip the learning curve and not pay the tuition yourself, sit down with someone who has already done it. That, honestly, is exactly where I am. You know where to find me.

Good luck. You're at the start of something good.

From user to a business that knows AI.

Bonus: The Complete Claude Toolkit

This is the toolbox that comes with the book. No more theory, but things you can copy and adapt today: ready-made Skills, a prompt library of more than thirty filled-in prompts, a starter template for project instructions, an AI policy on one page, a data-safety checklist, a connector shortlist, and a glossary. Everything is filled in with realistic examples. Wherever you see *italic text in brackets*, replace it with your own details. The rest can stay.

Don't use this part as reading material but as a reference. Take what you need, paste it into Claude, adapt it, and build your own library that way. Because that's the through-line of the whole book: the more you make it your way of working, the more it delivers.

Block 1: ready-made Skills

A Skill is a folder with a text file (`SKILL.md`) that teaches Claude how to do a recurring job your way. You write it once, and after that Claude picks it up automatically as soon as the task comes by. There are roughly two kinds. The first gives Claude an ability it didn't have without the Skill (making a PDF, testing a website). The second, and for small business by far the most important, captures your way of working: how you build a quote, how you answer a complaint, how your brand sounds. Claude can already do those tasks, but the Skill keeps it from guessing every time.

Three things make a Skill good. The description is sharp, because that's how Claude decides when to use the Skill. The instructions are concrete. And most important of all: give real examples, not made-up ones. A Skill with three real examples of your work is ten times better than one with only rules. Think of it as a new colleague you let watch a few times before they do it themselves. A rule of thumb from practice: a good Skill quickly saves forty percent of the time on that task, and keeps the quality consistent.

Below are five Skills useful in almost any small business. Copy the content, save it as `SKILL.md`, and add it through your Skills settings (in Claude.ai, Cowork, or Claude Code).

Skill 1: your brand voice

The most important Skill to start with. It makes sure every text — email, post, quote — sounds like your business and not like a generic robot.

```
---
name: brand-voice
description: Write and rewrite text in the house style and tone of
  [Company Name]. Use for emails, social posts, quotes, web copy, and
  anywhere the voice needs to be consistent.
---

# The voice of [Company Name]

## Who we are
[Company Name] is a [type of business] in [city] that does [what you do] for
[audience]. Our core values: [e.g. down-to-earth, helpful, expert].

## Our tone
- We write [informal but professional].
- Short, active sentences. No jargon, no buzzwords.
- Do: [e.g. concrete examples, dry humor in moderation].
- Don't: [e.g. exclamation points, "revolutionary," "unique," hype].

## Three real examples of our voice
EXAMPLE 1 - customer email:
"[paste a real email here that sounds exactly right]"
EXAMPLE 2 - social post:
"[paste a real post here]"
EXAMPLE 3 - quote intro:
"[paste a snippet that hits your voice]"

## Fixed phrasings
```

```
- Greeting: [Hi {first name},]
- Sign-off: [Best, {name} on behalf of [Company Name]]
- Words we never use: [list]
```

The three examples are the heart of this Skill. Take your time with it and pick texts that truly landed.

Skill 2: the quote builder

```
---
name: quote-builder
description: Draft a quote following the fixed structure of [Company Name].
  Use when I ask to create a quote, proposal, or estimate based on customer
  and project details.
---

# Quote builder [Company Name]

## Fixed structure
1. Short intro: understanding of the customer's question (2-3 sentences).
2. What we deliver: scope in clear language, with a bulleted list.
3. What we don't deliver: explicitly out of scope (prevents friction later).
4. Investment: [fixed price / day rate of $ ...]. Always plus applicable tax.
5. Timeline and schedule.
6. Terms: [payment within ... days, valid for ... weeks].
7. Close with a clear next step.

## Tone
Follow the "brand-voice" Skill. Confident, clear, no jargon without
explanation. Sell the outcome, not the hours.

## Ground rules
- Never invent prices or scope. Missing information? Ask your questions
  first and fill in nothing based on assumptions.
- Flag every assumption I need to check with >> NOTE <<.
```

Skill 3: the customer reply

```
---
name: customer-reply
description: Draft a reply to an incoming customer question or complaint for
  [Company Name]. Use for email, contact form, or review. The draft always
  passes a human before it goes out.
---

# Customer reply [Company Name]

## Approach per message
1. First briefly acknowledge the feeling or situation (1 sentence, sincere).
2. Give the concrete answer or solution.
3. Any next step or timeline.
4. Friendly, human close.

## Tone
Follow the "brand-voice" Skill. With a complaint: calm, take ownership,
don't get defensive. Never defensive, never a canned brush-off.

## Limits
```

```
- Never promise anything I haven't confirmed (refund, date, goodwill).
  Put such points as an option and flag them with >> CHECK <<.
- On legal threats or damage: don't answer yourself, escalate to me.
- Don't add apologies that admit fault without my approval.
```

Skill 4: the content repurposer

One good piece becomes five. This Skill gets the maximum out of what you already have.

```
---
name: content-repurposer
description: Turn an existing piece (blog, newsletter, transcript, customer
  call) into multiple social formats for [Company Name]. Use on "repurpose
  this," "make posts from this," or "chop this up."
---

# Content repurposer [Company Name]

## From 1 source to 5 pieces
Deliver by default, unless I ask otherwise:
1. One LinkedIn post (hook in line 1, 1 core idea, soft CTA).
2. Three short standalone tips or quotes (postable on their own).
3. One short script for a 30-60 second video.
4. One newsletter paragraph with a click-through.
5. Five subject lines / titles to choose from.

## Ground rules
- Follow the "brand-voice" Skill.
- Don't invent facts, numbers, or quotes that aren't in the source.
- One idea per post. No fluffy intros.
- Keep it in English, unless the source is in another language.
```

Skill 5: the monthly report

```
---
name: monthly-report
description: Summarize raw numbers or an export into a short, readable monthly
  report for [Company Name]. Use when I give a table, export, or loose numbers
  and ask for a report or summary.
---

# Monthly report [Company Name]

## Fixed structure (max 1 page)
1. Header: the three most important numbers of the month.
2. What went well (2-3 points, with a number attached).
3. What disappointed (2-3 points, honest).
4. Notable / explanation (where does an outlier come from?).
5. Recommendation: 1-3 concrete actions for next month.

## Ground rules
- Calculate only with the numbers I give. Don't make anything up.
- Separate fact from interpretation: put assumptions apart under "Assumption."
- Write for a busy owner: short, concrete, no jargon.
```

Where do you put a Skill? In Claude.ai and Cowork through your Skills settings; in Claude Code as a folder under `.claude/skills/`. The best part: a Skill works the same everywhere, so you build it once and share it with your whole team.

Skills that already exist (borrow or recreate)

You don't have to build everything yourself. There are now thousands of ready-made Skills, official and from the community. The official ones (from Anthropic) are the safest; community Skills can run code, so install only what you trust and look inside first. A selection of what we came across, to borrow or recreate:

- **Official (Anthropic):** the Document Suite (actually make Word, Excel, PowerPoint, and PDF, not just read them), Frontend Design (cleaner interfaces), Brand Guidelines or Theme Factory (your brand automatically on every artifact), MCP Builder (have a connection built), Skill Creator (make Skills yourself), and more playful ones like Algorithmic Art and the Slack GIF Creator.
- **Marketing and content:** the Marketing Skills from Corey Haines (copywriting, conversion optimization, SEO, launches, pricing — a set of about 25), a Branded Carousel Generator, and a Brand Design System.
- **Sales and strategy:** a Lead Research assistant, a Business Case Builder, and strategy Skills around well-known frameworks like Jobs-to-Be-Done and StoryBrand.
- **Operations and office:** an Invoice Organizer, a File Organizer, a Meeting Insights analysis (pull action items from notes), and Internal Comms.
- **Building and tech:** obra's Superpowers set (with /brainstorm, /write-plan, /execute-plan, and Systematic Debugging) and Webapp Testing with Playwright.

Anthropic even released a set of more than thirty Skills specifically for small businesses, with gems like a "Friday business pulse" (a Friday summary from your calendar, email, and bookkeeping), a job-posting builder, and an analysis that estimates per customer how well they pay and drafts a reminder right away.

Where do you find them? Start with the official examples and the big collection lists on GitHub (see the sources in Block 7). The rule of thumb: borrow the idea and the structure, but pour your own examples and ground rules into it. A Skill is only truly yours when it reflects your work.

Block 2: the prompt library

Before the individual prompts come, three structures that always work. Remember these and you won't need most prompt courses.

The first is the nine-point formula, the sturdiest recipe for an important task. Fill in what you need and leave out the rest:

```
You are a [role] who helps [audience] achieve [goal].
Use this input: [what you provide].
Respect these limits: [e.g. length, what not, tone].
Follow these steps: [1, 2, 3].
Write in [tone and style].
Deliver as [format: email / list / table / one-pager].
Meet this quality bar: [when is it good?].
```

The second is the CRISP mnemonic for quick prompts: Context, Role, Instructions, Specifications, Perspective. Almost every weak prompt is missing one of those five.

The third isn't a recipe but a habit, and maybe the most useful of all. Have Claude ask questions first:

```
Before you answer: ask me the questions you need to do this really
well. Only start once you know enough.
```

And as a second standing habit, to keep yourself sharp:

```
Red team this: [paste your plan, text, or idea]. What's wrong with it?
What are the weaknesses, risks, and failure scenarios? Be specific
and don't spare me.
```

Below are more than thirty filled-in prompts, by role. Replace the *italic parts*.

For leadership and strategy

```
Make a SWOT for [my company in 1 sentence] and then pick the three most
important priorities for the coming quarter. Justify the choice.
```

```
I'm considering [investment / new tool / new service]. Weigh the pros and
cons for a company of [size] in [industry]. Separate fact from assumption
and end with a recommendation plus the three questions I should answer first.
```

```
Summarize these meeting notes into one page: decisions, action items with
owner, and open points. [paste notes]
```

```
I need to communicate [difficult decision] to [team / customer]. Write three
versions: businesslike-short, empathetic, and very direct. [context]
```

```
Play devil's advocate on my plan to [plan]. Give the three strongest
counterarguments and what I'd need to see to change my mind.
```

For sales

```
Research [company name + website]. Give: what they do, three likely pain
points that [my service] addresses, and an opening line for a conversation.
Separate facts from assumptions.
```

```
Write a cold email to [name, title] at [company]. One concrete pain
([pain]), how [my product] solves it with [feature], and one clear CTA:
[action]. Max 120 words. Then five subject lines with a curiosity hook,
without the words "quick" or "ask."
```

```
Draft a quote for [customer] for [project/scope] at [price/rate]. Follow my
fixed quote structure. Flag assumptions.
```

```
Come up with the five most likely objections to [offer] and write a short,
honest response to each that applies no pressure.
```

```
Write a follow-up email after my conversation with [name] about [topic].
Summarize what we agreed, restate the value, propose the next step.
```

```
Analyze [company]'s news from the past few months and build a "why us, why
now" angle. Separate confirmed facts from assumptions so I don't over-claim.
```

For marketing and content

```
Make a content calendar for [month] for [company/industry]. Eight ideas,
mixed: educational, behind-the-scenes, customer story, offer. Per idea a
title and the format (post / video / newsletter).
```

```
Write a LinkedIn post about [topic]. Hook in the first line, one core idea,
a soft call-to-action. Follow my brand voice.
```

```
Repurpose this into five pieces (post, three standalone tips, short video
script, newsletter paragraph): [paste blog/transcript].
```

```
Give ten long-tail keywords around [product/service] with good potential for
local visibility in [city/region], with a meta description of max 160
characters per keyword.
```

```
Write three ad variants for [offer], each with a different hook (problem,
result, curiosity). Audience: [who].
```

```
Write a customer story that shows how [product/service] made the difference
for [type of customer]. Problem, approach, result. No invented numbers; leave
blanks where I fill in real figures.
```

For operations, admin, and finance

```
Turn these loose steps into a clean work instruction (SOP) a new colleague
can follow, with a short checklist at the end: [paste steps].
```

```
Analyze this export and give the three most important insights plus something
I'm likely missing. Calculate only with these numbers and name every
assumption separately. [paste table/CSV]
```

```
Summarize these numbers into a one-page monthly report following my fixed
structure: [paste data].
```

```
Write a friendly payment reminder for [customer], invoice [number], [amount],
[days] overdue. Then a second, slightly firmer version for if the first gets
no response.
```

```
Help me find the biggest time-sink in this process and propose two ways to
shrink it: [describe process step by step].
```

For service and support

```
Draft a reply to this complaint. First briefly acknowledge the feeling, then
give the solution. Promise nothing I haven't confirmed; flag such points.
[paste message]
```

```
Make a FAQ of ten questions based on these common tickets, with short,
friendly answers: [paste examples].
```

```
Analyze these reviews: top three compliments, top three complaints, and the
through-line. Proposal: what do we improve first? [paste reviews]
```

```
Turn this resolved ticket into a knowledge-base article so a colleague can
solve it themselves next time: [paste ticket].
```

```
Categorize these incoming messages into [e.g. question, complaint, sales,
spam] and propose the right next action per category. [paste list]
```

For yourself: productivity and learning

```
Explain [difficult topic] as if I know nothing about it, with a real-world
example. Then ask me two questions to check that I get it.
```

```
Summarize this long document into the five decision points that truly matter,
and say what I'd miss if I only read the summary. [paste document]
```

```
Rewrite this email so the tone is friendly but confident, without me giving
in too much: [paste draft].
```

```
I want to get better at [skill]. Ask me questions via the Socratic method
instead of giving the answer, so I figure it out myself.
```

```
What am I missing? Here's my approach to [task/plan]. Name my blind spots and
the assumptions I haven't made explicit.
```

Block 3: project instructions and power tricks

A Project (or in Claude Code a `CLAUDE.md` file) is the memory that travels to every conversation. Here you set down who you are, how you work, and what the ground rules are, so you don't have to type it again every time. Below is a filled-in starter template you paste into the project instructions.

```
# About us
We are [Company Name], a [type of business] in [city]. We do [what] for
[audience]. Work in English by default.

# How I want you to communicate
- Direct and concrete, without disclaimers.
- Short where you can; detailed where you must.
- When in doubt: ask one sharp question before you start.
- Disagree? Say so, with reasoning. I don't want a yes-man.

# Ground rules (always)
- Never invent numbers, sources, names, or quotes. If you don't know,
  say so.
- Separate fact from assumption. Flag assumptions I need to check.
- Output that goes to a customer passes me first.
- Never paste customer or personal data into examples.

# Our voice
[Two lines about tone, or: "follow the brand-voice Skill."]

# Fixed facts you may know
- Services: [list]. Don't do: [what's out of scope].
- Important terms with us: [term = meaning].
```

The golden rule for what belongs in here comes from experienced users: if you want Claude to still know this in six months, put it in the project instructions; if it might change next week, use a regular prompt. So don't put sprint goals in here, no fast-changing prices, and — very important — never passwords, API keys, or sensitive data. In Claude Code such a file is often shared via Git, and then your secret is suddenly online. Keep it short and scannable; fifty to two hundred lines is a fine guideline. In Claude Code you can add something to the memory in one line with the # sign, and create a starter file with `/init`.

Beyond that, a handful of tricks advanced users use to make the difference:

- **Plan first, build second.** On a big job, ask Claude for a plan first and approve it before it begins. In Claude Code you turn on plan mode for that (shift+tab). This keeps it from running the wrong way.
- **Make your own slash commands.** Save a recurring prompt as a command (in Claude Code under `.claude/commands/`), so you call it up with one word.
- **Let heavy work be done by subagents.** Digging through log files or doing research? Let a subagent do it in its own window and return only the conclusion. That keeps your main conversation clean and sharp.
- **Skill or project instruction?** If a rule applies to almost every task, put it in the project instructions (always active). If it's a specific workflow that only sometimes applies, make it a Skill (loads only when needed). That way you don't waste room.

- **Use a handoff file.** Long project running, or a conversation filling up? Have Claude write a `handoff.md` with what was tried, what worked, and what didn't. Start a fresh conversation with only that file and you continue seamlessly.
- **A plugin is a toolbox.** It bundles Skills, commands, subagents, and connectors in one install. Handy when you want to share a complete way of working with your team.

Block 4: AI policy on 1 page, plus data checklist

An AI policy isn't legally required, but it prevents trouble and gives you a legal handle if something goes wrong. You don't need a legal department: an owner plus someone from HR or finance is enough. Start by inventorying which tools your people already use — that's often surprisingly many, and immediately your list of approved tools. Below is a filled-in template that fits on one page.

```
AI POLICY [Organization] - version 1.0 - [date] - review: [date + 1 year]

1. PURPOSE
Encourage responsible and safe AI use, limit privacy and liability risks,
and comply with laws and regulations.

2. WHO IT'S FOR
All employees, freelancers, interns, and outside parties with access to
our systems or data.

3. WHAT WE MEAN BY AI
All AI tools, including built-in ones (e.g. Copilot, smart email features,
meeting summarizers). Not just standalone chatbots.

4. USE - THREE CATEGORIES
Allowed without checking: [e.g. improving text, brainstorming,
  summarizing non-sensitive material, making drafts].
Prohibited: [entering customer/personal data, legal or medical decisions
  without review, sending output unseen to the outside].
With permission: [adopting a new tool, AI in a customer process,
  automated sending].

5. DATA
Never put "red data" in an unapproved tool (see checklist below).
For work, use only the [approved, business] tier.

6. HUMAN CONTROL
AI delivers a draft; a human decides and is responsible. Output to
customers is always reviewed.

7. TRANSPARENCY
We're honest with customers where required (e.g. with a chatbot), and we
don't pass off AI-generated reviews or endorsements as real.

8. OWNER AND REPORTING POINT
Questions or incidents: [name / email].

9. REVIEW
Reviewed at least annually, and as soon as the law or our tools change.
```

Alongside the policy belongs a **data-safety checklist**. This is "red data" you never paste into a free or consumer AI: personal data (national ID or BSN, passport, ID card), login credentials, passwords and API keys, customer data (names, email addresses, account numbers), financial figures and prices, source code, contracts and NDAs (pasting an NDA into a chatbot can itself be a breach of contract), medical information, and your own strategy or roadmap.

The most important rule almost no one knows: there's a world of difference between the free and the business version. On the **consumer or free tier**, your input can be used to train models. The **business tiers** — Claude for Work/Team, ChatGPT Enterprise, Microsoft Copilot with Commercial Data Protection, Gemini for Workspace — promise contractually that they will not use your data for training. Most employees have no idea which tier they're on. That this matters shows in the numbers: research from LayerX and Cyberhaven shows that 77% of employees have at some point put sensitive company data into an AI tool, and that the share of sensitive data in what people paste rose in a year from nearly 11% to over 27%. Zscaler counted more than 410 million data-leak signals via ChatGPT alone in 2025. The simple countermeasure: use the approved tier and remove personal data before you paste anything.

The EU AI Act and GDPR: what an SME should practically know

No panic, but worth arranging. The core for an SME:

- **One duty already applies**, since February 2025: AI literacy (Article 4). Anyone who works with AI must understand, at a basic level, what it does and where the risks are. This book plus a short internal briefing largely covers it.
- **Four risk levels.** Most SME uses fall under "minimal" or "limited" risk. Limited risk mostly means a transparency duty: tell people they're dealing with AI, and label AI content. High risk (think CV screening or credit scoring) is heavy, but rare in an SME — worth a quick check.
- **Key date: 2 August 2026**, when the transparency rules (Article 50) and the SME provisions start to apply. For small companies there's simplified documentation and lower cost. (A proposal, the "Digital Omnibus," may ease some deadlines — keep an eye on it.)
- **A workable step plan** that costs almost nothing: inventory your AI tools, decide the risk level per tool, train your team (Article 4), write this AI policy, arrange the transparency, and keep the evidence (inventory, training record, signed policy) in one shared folder. Done.
- **GDPR still applies on top of this** for personal data: a lawful basis, data minimization, customer rights, and a Data Processing Addendum with any provider that processes personal data for you (chapter 7).

(Selling into the US instead? There it's a patchwork of state privacy laws (CCPA and others) and FTC rules rather than the EU AI Act and GDPR; the US edition of this handbook covers that.)

Block 5: connector shortlist for small business

A connector (technically: an MCP connection) links Claude to a system you already use, so it no longer has to work with separate copy-paste actions. MCP is an open standard Anthropic introduced in 2024 and which is now carried by the whole market — by early 2026 there are already more than ten thousand ready-made connections. You only need a few. The art is: connect what fits your real work, not everything at once.

Connector	What it gives you	Where to start
Calendar (Google/Outlook)	Schedule appointments, read availability	The first connection for most
Email (Gmail/Outlook)	Drafts, follow-up, summarize threads	Start read-only
Cal.com	Have appointments booked automatically	If you schedule a lot of calls or demos
Slack / Teams	Messages, summaries, notifications	For internal communication
Notion / Drive / SharePoint	Unlock your documents and knowledge base	Connect your process documents
CRM (HubSpot/Salesforce/SAP)	Read, enrich, fill customer data	See the sales chapter; minimal permissions
Bookkeeping (Exact/QuickBooks/ Xero)	Summarize and report numbers	Start with a read-only export
Shopify / web store	Products, orders, inventory	For e-commerce
Payments (Stripe/Mollie)	Pull payment status and invoices	Start read-only
Email marketing (Mailchimp/ Klaviyo)	Campaigns, lists, stats	For newsletters
Ads (Google/Meta Ads)	Pull campaigns and numbers	For anyone advertising
Project management (Linear/Jira/ ClickUp/Monday)	Tasks, status, sprints	Where work is tracked
Web search (Brave/Tavily)	Current info from the web, with source	Against outdated knowledge
Browser (Playwright/ Browserbase)	Operate and read sites	For repetitive work online

Connector	What it gives you	Where to start
Design (Figma/Canva)	Pull or make designs and assets	For marketing and design
Your own database	Query and update your own data	See "Which database?" below
Aggregator (Composio/Zapier)	Hundreds of apps through one connection	When you want broad reach fast

Two rules keep it safe. Give a connector as few permissions as possible: read-only where you can, and access only to what's needed. And keep visibility on what happens through logs. That way the agent reaps the benefits of your systems without you losing control. A good test for every choice: if a better model comes out next month, can I use it without rebuilding my whole setup? With an open standard like MCP, the answer is yes.

Which database fits your context?

Often you don't need a database at all. For reference — FAQ, prices, policy — a Project's knowledge base is enough. A database gets interesting only when you have structured data Claude needs to look up or update: customers, products, inventory, orders. Then there are roughly three levels, rising in power.

For most small businesses a **no-code database** is the sweet spot: Airtable, Notion, or even Google Sheets. No programmer needed, it works like a smart spreadsheet, and you connect it through a connector. Airtable is strongest as a real relational database (linked tables, many views); Notion shines when you want structured data alongside documents; Google Sheets is the lowest-barrier because everyone already has it. Keep in mind the per-user price adds up.

Outgrow this — more data, real queries, multiple users — and a full database is the next step, and the most accessible is **Supabase**, a managed version of PostgreSQL (the most-used open-source database). Important for you: Supabase has been an official Claude connector since early 2026, it comes with an official "Postgres best practices" skill, and it respects your access rights per row. The advantage over no-code: Claude "speaks" the language of a database (SQL) natively, so there are fewer layers between your question and the answer where something can break. Free to start, paid from around €25 a month.

Want Claude to be able to search by meaning across a big pile of documents (semantic search, also called "RAG")? Then you often hear you need a separate "vector database," like Pinecone or Qdrant. For most small businesses that's overkill. With pgvector — an extension built into Supabase and Postgres by default — you store those meaning-fingerprints right in your own database: documents and search index in one table, one query, nothing to sync. That's fine up into the millions of items. You add a separate vector database when you truly reach that scale.

Three rules of thumb to close: pick the simplest option that fits (start with your knowledge base, then no-code, then Postgres); give the connection minimal permissions (read-only where you can); and keep your data exportable, so you're not locked in anywhere. That's how you build a context store that grows with you without committing you to one vendor.

Block 6: how you build your company's context

In part 1 we saw the most important lesson of all: context beats the prompt. This block goes a step further, because it's exactly where the most gain sits. The question isn't "how do I write a clever prompt" but "how do I teach my AI my way of working." Now that everyone has the same models, your edge no longer sits in the model, but in what the model knows about your business. A sharp saying from the field: "if my model just knew what my company knows, we'd be a hundred times more productive." Turning your definitions, your fixed processes, your decision rules, and your exceptions into something an AI can read — that's the real work. And the nice part: context isn't a prompt you throw away, it's an asset that compounds. A good prompt gives you one good answer; good context makes sure the thousandth answer is still right.

Where your context lives

You don't have to build a new system. You've already met the building blocks in this book; the art is to see them as one whole. Give each kind of knowledge a fixed place:

- **Who you are, your voice, and your ground rules** belong in your project instructions (Block 3).
- **How you do something, your fixed processes** belong in Skills (Block 1).
- **Reference** — prices, FAQ, product details, policy — belongs in your knowledge base or Project (part 4).
- **Living data** from your own systems comes in through connectors (Block 5).
- **The glue** is the corrections and examples you add along the way.

That way not everything lands in one unwieldy prompt, but you know exactly where what belongs — and you can maintain it.

The hardest step: getting knowledge out of heads

Your most valuable knowledge isn't written down anywhere; it's in your people's heads. Three ways that work in practice. The first: record yourself while you do it. Perform a task once thinking out loud, with a screen recording or just your voice, and have Claude turn it into a draft Skill or work instruction. You correct it, and in half an hour you've captured what otherwise stayed elusive. The second: have Claude interview your best employee. Give the instruction "ask me questions about how I build a quote, and keep probing for the why" and have it summarize the answers into a process. The third: mine what you already have. Old emails, resolved tickets, and your best quotes are gold mines. Have Claude distill the patterns and fixed steps from them — not dump the loose documents, but pull out the way of working.

Build from use, not up front

Don't start with a big documentation project that never gets finished. Start small and let your context grow out of real use. The habit that makes the difference: every time you have to correct the AI, capture that correction — as a rule in your project instructions or as an

example in the Skill. Every fix makes the system lastingly better. That way your context becomes, on its own, an ever-sharper portrait of how your company works, without it ever becoming a separate project.

Less, but sharper

Here's a counterintuitive truth: more context isn't better. Too much or messy context actually worsens the answers. Professionals call that "context rot" — quality drops as the window fills with noise, even before you hit technical limits. A small, well-chosen context beats a large one full of unnecessary stuff. So don't put in what the model already knows (general knowledge), and not what has nothing to do with the task. Context goes wrong in four ways, in plain language: missing (it doesn't know something), outdated (an old price or expired policy), contradictory (two sources disagree), and too much noise. The remedy is always the same: clean up and keep one source of truth.

Who owns it

Maintaining context is teamwork. One good project instruction from you is nice, but the companies that truly lead make it a shared habit: one place that is the truth, an owner per part, and a fixed moment to update. Make "does this change our way of working? Then update the context" a standard question with every process change. And schedule half an hour each quarter to go through your project instructions and Skills: what's no longer correct, what's missing, what can go.

A starter plan in five steps

Pick one process that comes back often and causes irritation. Capture the current way of working — through recording, interviewing, or mining — and have Claude turn it into a draft Skill or project instruction. Use it for a week and correct it, with every correction going back into the context. Add the right connector as soon as manual copying becomes the bottleneck. Finally, assign an owner and put a quarterly review on the calendar. After that you take on the next process.

Do this a few times and you no longer have loose tricks, but a business that has taught its own way of working to AI. That's the edge a competitor can't simply copy — because it sits in your context, not in the model everyone can buy.

Block 7: glossary and sources

A short glossary of the terms you've come across in this book, in plain language.

Term	In plain language
Prompt	The instruction or question you give Claude.
Context	Everything Claude "knows" while doing your task: instructions, documents, data, history.
Project instructions	Fixed ground rules and facts that travel to every conversation within a Project. In Claude Code: the CLAUDE.md file.
Project	A workspace with its own instructions and a knowledge base, so Claude remembers your context.
Knowledge base	The documents you attach to a Project as reference (prices, FAQ, policy).
Skill	A reusable package (SKILL.md) that teaches Claude your way of a recurring task.
Connector (MCP)	A safe link between Claude and a system you already use (calendar, CRM, email).
Subagent	A helper Claude with its own window that does a subtask and returns only the result.
Plan mode	Claude makes a plan first and waits for your approval before it begins.
Plugin	A bundle of Skills, commands, and connectors you install at once.
Hook	An automatic action that fires at a fixed moment (for example a check on delivery).
Context window	The amount of information Claude can "hold" at once.
Context engineering	The craft of supplying the right context at the right moment, instead of just clever phrasing.
Hallucination	When AI confidently makes up something that isn't true. Always verify.
Consumer vs business tier	The free version can train on your input; business versions promise contractually that they won't.
AI literacy	Basically understanding what AI does and where the risks are. Required under the EU AI Act (Article 4), in force since February 2025.
Shadow AI	Employees using AI tools without any agreements, often with company data. The biggest leak.

Sources for further reading

Official (Anthropic): the product documentation at docs.claude.com and for Claude Code code.claude.com; the power-user tips in the Help Center; and the excellent article Effective context engineering for AI agents. The open standards: modelcontextprotocol.io for connectors and agentskills.io for Skills.

Community (collections on GitHub, quality varies — use at your own risk): anthropics/skills (official examples), ComposioHQ/awesome-claude-skills and travisvn/awesome-claude-skills for Skills, langgptai/awesome-claude-prompts for prompts, hesreallyhim/awesome-claude-code for Claude Code, and obra/superpowers for a complete way of working.

EU SME: the European Commission's AI Act hub for the rules and SME guidance, your national data-protection authority and the EDPB for GDPR, and for the Netherlands the KVK guide to an AI policy. For an AI policy template, several EU bodies and your trade association publish SME-friendly versions.

Finally

This is where the toolkit ends and your work begins. Copy one Skill, paste one prompt, fill in the AI policy, or capture one process — but do something. Because as the whole book showed: you don't learn AI from a book, you learn it by doing it. And every time you capture something, you build toward a business that has taught its own way of working to AI. No competitor can get around that.

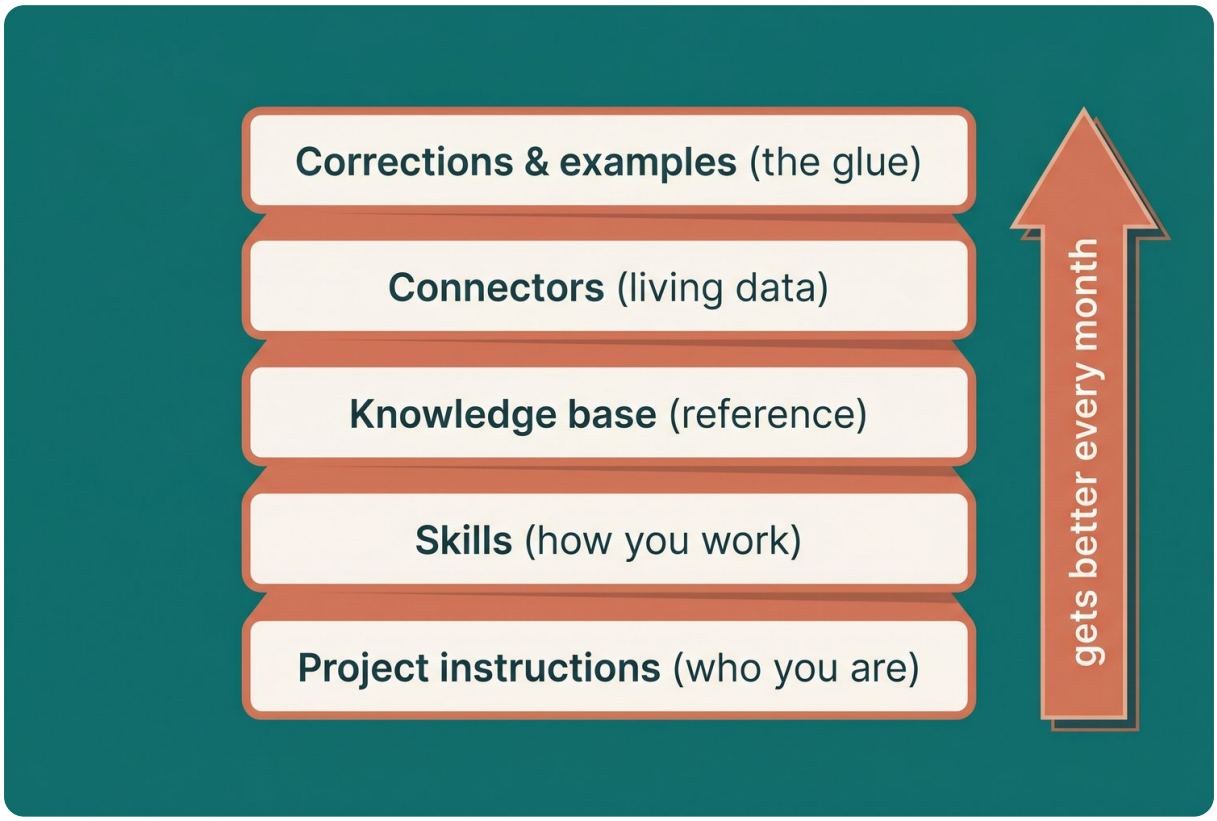


Figure - the "context stack" as stacked layers from bottom to top: project instructions (who you ar