FEBRUARY 2026 EDITION

Ofir's Claude Code Field Manual

A practical guide for non-developers who want to build real things — dashboards, websites, automations — without writing code or waiting for IT.

Ofir Bloch | 10 Project Playbooks | Zero Code Required

ABOUT THIS GUIDE

What's inside (and who it's for)

This guide is for marketing people, customer success managers, sales ops leads, founders, and anyone else who has ideas and projects but no programming background. You've been told you need a developer to build things. As of early 2025, that changed. Now it's mostly not true.

Claude Code is a tool that builds things when you describe what you want in plain English. Websites. Dashboards. Automated reports. Internal tools. The kinds of things that used to require a developer, a sprint planning meeting, and six weeks of "it's in the backlog."

This guide starts with the easy stuff (Cowork, which works like chatting with a very competent assistant) and progresses to the powerful stuff (Claude Code Desktop, which actually builds and deploys real software). You don't need to know what a terminal is. You don't need to know what "git" means. I'll explain every term the first time it shows up.

CONTENTS

No coding experience required. Seriously.

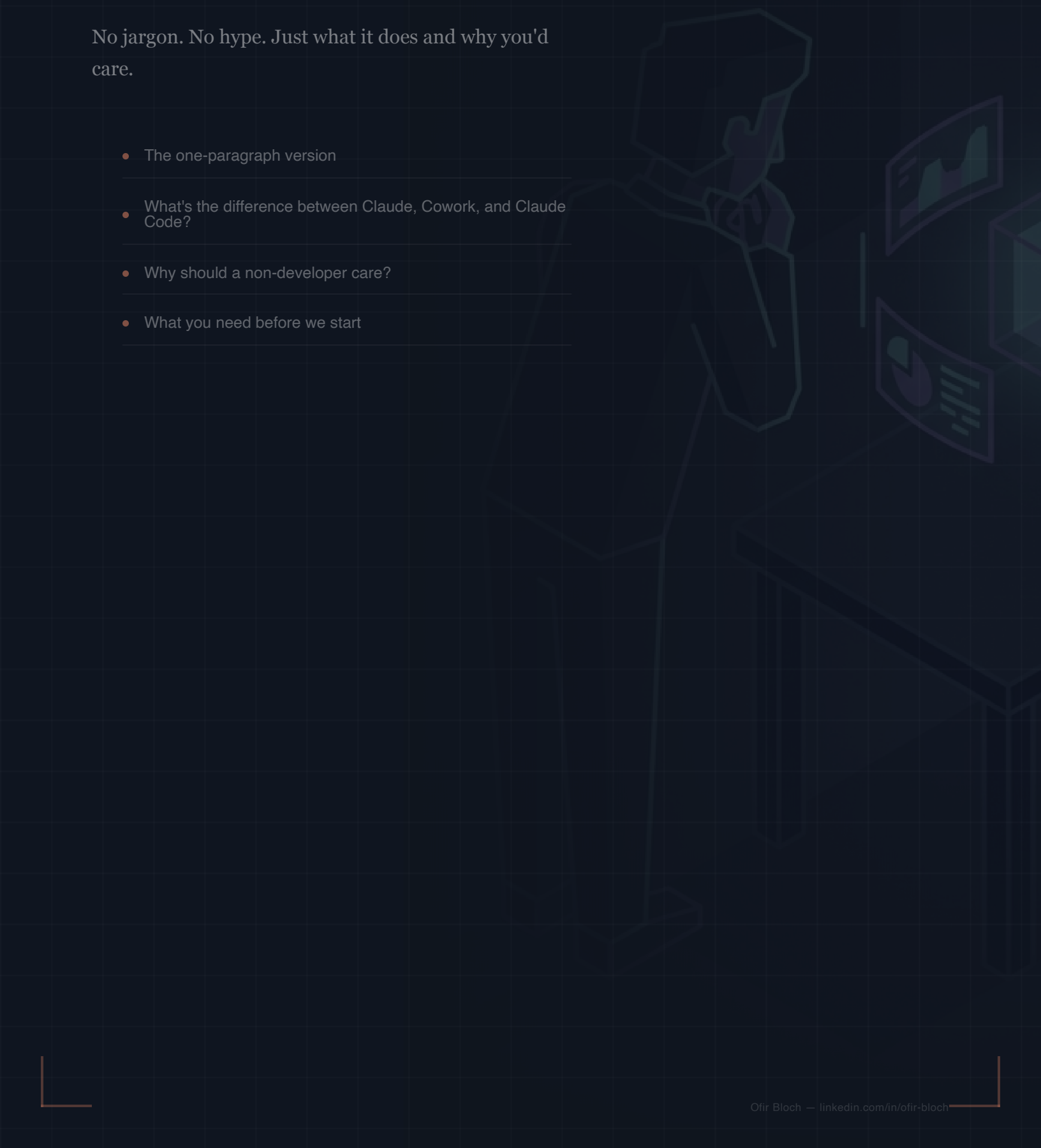
Table of Contents

PART ONE

What Is This Thing?

No jargon. No hype. Just what it does and why you'd care.

- The one-paragraph version
- What's the difference between Claude, Cowork, and Claude Code?
- Why should a non-developer care?
- What you need before we start



PART 1 — WHAT IS THIS THING?

The one-paragraph version

Anthropic (the company that makes Claude, the AI you might already chat with) built a tool called Claude Code. It does more than answer questions. It can actually build things: websites, apps, dashboards, automated workflows, internal tools. You describe what you want in plain English, and it writes the code, creates the files, tests them, and can even put the finished product on the internet. You don't write code. You describe outcomes.

What's the difference between Claude, Cowork, and Claude Code?

Think of them as three levels of the same AI:

	CLAUDE CHAT	COWORK	CLAUDE CODE
What it is	AI chatbot	Claude with hands	Claude with a workshop
What it does	Answers questions, writes text, analyzes documents	Creates files, builds spreadsheets, processes data on your computer	Builds full apps, websites, dashboards, deploys to the internet
What it can't do	Touch your files or build anything	Put things on the internet or connect to databases	Nothing major — this is the full toolkit
Who it's for	Everyone	Anyone who works with documents and data	Anyone who wants to build real tools

This guide covers Cowork and Claude Code. You'll start with Cowork (easy, low-risk, useful immediately) and graduate to Claude Code (powerful, slightly more involved, life-changing for your team's velocity).

Why should a non-developer care?

Because the bottleneck in your work has never been ideas. It's been execution. You know exactly what dashboard your team needs. You know the landing page that would convert better. You know the weekly report that should be automated. But every one of those things requires "dev resources." And dev resources are always allocated to something else.

Claude Code changes the math. Not for everything. But for a surprising number of the things that are stuck in someone's backlog, you can now build them yourself. In hours, not sprints.

REALITY CHECK

Claude Code won't replace your dev team. But it will let you stop waiting for things that don't actually need a developer. That distinction is worth understanding.

What you need before we start

- A computer (Mac or Windows). Chromebook works for Cowork only.
- A Claude account on a paid plan. Pro ($20/month) works. Max ($100/month) is better if you'll use this daily.
- The Claude Desktop app installed. Download it at claude.ai/download.
- Patience for about 30 minutes of setup.
- No coding experience. No terminal experience. No git experience. None of that.

What you'll learn in this guide

- **Part 2** — How to use Cowork for everyday tasks (your comfort zone)
- **Part 3** — How to use Claude Code Desktop to build real projects
- **Part 4** — How to communicate with Claude so it actually delivers
- **Part 5** — The safety guardrails you need to know
- **Playbooks** — 10 step-by-step projects you can build this week
- **Part 6** — What to do when things go wrong

PART TWO

Start Here: Cowork

Your comfort zone. No code. No terminal. Just results.

- What Cowork is
- How to open Cowork
- Your first task: messy doc to clean report
- Ten things Cowork is great at
- Cowork limitations

PART 2 — START HERE: COWORK

What Cowork is

Cowork is a feature inside the Claude Desktop app. You open it by clicking the "Cowork" tab at the top of the app. It looks like a regular Claude chat, but it can do things regular Claude can't: access files on your computer, create documents, build spreadsheets with working formulas, run long tasks in the background, and work on multiple things at once.

JARGON BUSTER

Sandbox — A secure, protected area on your computer where Cowork does its work. It can't accidentally delete your important files or do anything destructive. Think of it as a playpen for AI.

How to open Cowork

1. Open the Claude Desktop app (the orange icon in your dock or taskbar).
2. Look at the top of the app. You'll see tabs: **Chat** , **Code** , and **Cowork** .
3. Click **Cowork** .

That's it. You're in.

SAVE YOURSELF 45 MINUTES

If you don't see the Cowork tab, make sure your Claude Desktop app is updated. Go to claude.ai/download and install the latest version.

Your first task: turn a messy document into a clean report

Let's do something practical. Say you have a messy Google Doc full of notes from a customer call, and you need to turn it into a clean summary for your team.

1. In Cowork, drag the file into the chat area. Or click the paperclip icon and select it.
2. Type your prompt (feel free to copy and paste it).
3. Press Enter (or click the send button).
4. Cowork will read the file, extract the important parts, and create a formatted Word document.
5. When it's done, you'll see a download link. Click it to save the file.

COPY THIS EXACTLY

"Read this document. Extract the key points, decisions made, and action items. Create a clean, formatted summary document with sections for: Key Takeaways, Decisions, Action Items (with owners if mentioned), and Open Questions. Make it professional enough to share with my team."

Ten things Cowork is great at

These are real tasks non-developers use Cowork for every day:

1. **Clean up messy documents** — Turn raw notes into polished reports, briefs, or summaries.
2. **Build spreadsheets with formulas** — Describe the logic you need; Cowork writes the formulas.
3. **Analyze data from files** — Drop in a CSV and ask questions about it.
4. **Create presentations outlines** — Draft slide decks from research or meeting notes.
5. **Write and format long documents** — SOPs, handbooks, proposals with consistent formatting.
6. **Compare documents** — Find differences between contract versions or policy updates.
7. **Extract data from PDFs** — Pull tables, figures, and key data from PDF reports.
8. **Generate charts and visualizations** — Basic charts from your data, downloadable as images.
9. **Batch-process files** — Apply the same transformation to 50 files at once.
10. **Draft email sequences** — Write multi-touch campaigns with consistent voice.

WORTH KNOWING

Cowork can work on multiple tasks in the background. Start one task, then open a new chat thread and start another. It's like having several assistants working in parallel.

Cowork limitations (when to graduate to Claude Code)

Cowork is powerful, but it works with files. It can't put a website on the internet. It can't build something your team logs into every day. It can't connect to your company's database or pull live data.

When you hit one of these walls, that's when you move to Claude Code Desktop. Which is what Part 3 is about.

REALITY CHECK

Most people don't need Claude Code Desktop. If Cowork handles your needs, stay there. Simpler is better. Only graduate when you genuinely hit a wall.

PART THREE

Level Up: Claude Code Desktop

Where you start building things that live on the internet.

- What Claude Code Desktop is
- Opening it, step by step
- The three modes (Plan / Code / Act)
- The permission system
- The diff view
- Your first project: a landing page
- The 5-minute project setup
- Your project's memory: CLAUDE.md
- Running multiple sessions

PART 3 — LEVEL UP: CLAUDE CODE DESKTOP

What Claude Code Desktop is

Claude Code Desktop is the "Code" tab in the same Claude Desktop app you already have. When you click it, you get a different interface. Instead of chatting about files, you're working inside a project. Claude can create files, organize them into folders, run them, test them, and deploy them. It's the difference between writing a recipe (Cowork) and actually cooking the meal (Claude Code).

JARGON BUSTER

Project folder — A folder on your computer where all the files for one project live. Think of it as a filing cabinet drawer dedicated to one thing. Claude Code works inside this folder.

Opening Claude Code Desktop, step by step

1. Open the Claude Desktop app.
2. Click the **Code** tab at the top.
3. It will ask you to select a folder. This is where your project will live.
4. If you don't have a folder yet: On Mac, open Finder, go to Documents, right-click and choose New Folder. Name it something like **"my-first-project"** (no spaces, use dashes instead). On Windows, open File Explorer, go to Documents, right-click and choose New > Folder.
5. Select that folder in Claude Code Desktop.
6. You'll see a chat-like interface with your project name at the top. This is your workspace.

SAVE YOURSELF 45 MINUTES

Use dashes instead of spaces in folder names. "my-project" works. "My Project" might cause problems later. It's a small thing that prevents headaches.

The three modes

Claude Code Desktop has three modes. You switch between them using the dropdown in the chat area.

Mode	What it does	When to use
Plan	Claude reads your files and describes what it would build.	Always start here. Review before building.
Code	Claude creates and edits files, asks permission for each action.	After you've approved the plan. Your main building mode.
Act	Claude runs commands, installs tools, deploys. More autonomy.	Only when you're comfortable and have safeguards in place.

⚠ **DO NOT SKIP THIS**

Start in Plan mode. Always. Describe what you want. Let Claude explain how it would build it. Review that plan. Then switch to Code mode and say "Go ahead." This saves you from Claude enthusiastically building the wrong thing.

The permission system (your safety net)

In Code mode, every time Claude wants to create a file, change a file, or run a command, it shows you what it's about to do and asks: allow or deny? You get three choices:

OPTION	WHAT IT MEANS
Allow Once	Do this one thing, then ask me again next time.
Allow Always	Do this type of thing without asking for the rest of this session.
Deny	No. Don't do that.

This means Claude literally cannot do anything without your say-so. If you see something you don't understand, click Deny. Nothing bad happens. Claude will explain what it was trying to do and suggest an alternative.

WORTH KNOWING

When in doubt, use "Allow Once." You can always switch to "Allow Always" later once you're comfortable with what Claude is doing. Start cautious.

The diff view (seeing what changed)

After Claude makes changes, you'll see a "diff" view. This shows you exactly what was added (highlighted in green) and what was removed (highlighted in red). You don't need to understand the code. You're just looking for:

- **Does the amount of change seem right?** If you asked for a small fix and see 500 lines changed, something's off.
- **Are there any red lines that look important?** Claude sometimes removes things it shouldn't.

> **JARGON BUSTER**
>
> **Diff view** — A display that shows what changed in a file. Green lines were added. Red lines were removed. It's like Track Changes in Word, but for code.

If something looks wrong, you can undo it. Claude Code Desktop saves a snapshot before every change. Click the rewind button in the toolbar to go back.

> **THIS WILL BREAK THINGS**
>
> Never switch to Act mode unless you understand what your project contains. Act mode gives Claude more autonomy. It's for experienced users who have set up proper safeguards.

Your first project: a simple landing page

Let's build something real. A one-page website for a product, event, or team.

1. Open Claude Code Desktop. Select your empty project folder.
2. Start in **Plan mode**. Type your prompt describing the landing page you want.
3. Claude will describe the plan. If you see unfamiliar terms, type: "Explain [term] like I'm in high school."
4. Once the plan makes sense, switch to **Code mode**.
5. Type: "Go ahead and build it. After each major step, stop and show me what it looks like."
6. Claude will create files and tell you how to preview the site. Usually a link like http://localhost:3000. Click it.
7. Look at the preview. Tell Claude what to change.
8. When you're happy, say: "How do I put this on the internet so other people can see it?"

The 5-minute project setup

Before you build anything, do these six things. They take about five minutes and prevent the five most common mistakes: wrong folder structure, Claude forgetting your preferences, building without a plan, mixing up project files, and not knowing when to stop.

1. **Create your project folder.** Use dashes, not spaces: `my-webinar-page` not `My Webinar Page`. Spaces in folder names cause headaches.
2. **Open Claude Code in that folder.** In the terminal, type `cd my-webinar-page && claude`. Or in Claude Code Desktop, select the folder from the sidebar.
3. **Run /init** to generate a starter CLAUDE.md file. This is Claude's memory. We'll cover it on the next page.
4. **Create a .claudeignore file** to hide files Claude shouldn't touch (covered earlier in Part 2).
5. **Start in Plan Mode** (press Shift+Tab twice). Describe what you want to build. Let Claude ask you questions. Review the plan before it builds anything.
6. **Define "done."** Tell Claude what success looks like. "This project is done when: the page loads correctly, the form submits to our CRM, and it matches our brand colors."

SAVE YOURSELF 15 MINUTES

These 6 steps take 5 minutes. They save you from hours of frustration. Think of it as packing your bag before a hike. Nobody regrets being prepared.

MINI CHECKUP

- ☐ I have a project folder with a clear, dashed name
- ☐ I ran /init and customized my CLAUDE.md
- ☐ I have a .claudeignore file
- ☐ I started in Plan Mode
- ☐ I defined what "done" looks like

Your project's memory: the CLAUDE.md file

This is the single most impactful feature in Claude Code. CLAUDE.md is a plain text file that sits in your project folder. Every time you start a new session, Claude reads it automatically. Think of it as a sticky note on your desk that Claude checks before every conversation.

Without it, every new session starts from zero. You'd have to re-explain your brand colors, your preferred writing tone, your file naming preferences, every single time. With it, Claude already knows.

How to create one

Method 1: Type `/init` in Claude Code. It analyzes your project and generates a starter file. Then you edit it to add your preferences.

Method 2: Just ask Claude: "Create a CLAUDE.md file for this project that remembers my brand is X, my colors are X, and I prefer Y style."

GOLDEN RULE

If you change one thing about how you use Claude Code, make it this: create a CLAUDE.md file. Users report "night and day" quality differences.

SAVE YOURSELF 45 MINUTES

Without a CLAUDE.md, you spend the first 5-10 minutes of every session re-explaining your preferences. Over 5 sessions, that's 45 minutes of repeating yourself.

A starter template for non-developers

CLAUDE.md

```
# Project: [Your Project Name]

## What This Project Is
[One sentence: "A landing page for our Q3 webinar"]

## My Preferences
- Brand colors: [hex codes or color names]
- Tone of voice: [professional, casual, playful]
- My company name: [name]

## Important Rules
- Always ask before deleting any files
- Show me a plan before making changes
- Use simple, non-technical language

## Files That Matter
- [List key files as your project grows]
```

The "keep it short" rule

CLAUDE.md works best when it's under 150 lines. If it gets longer, Claude starts ignoring parts of it. Think of it like a one-page cheat sheet, not a manual. If you need to reference longer documents (like a full brand guide), point to the file instead of pasting the whole thing in.

When to update it

After every session where you had to correct Claude on something it should have known. If you told Claude "no, use our blue, not that blue" three times, add your blue hex code to the CLAUDE.md so it doesn't happen again.

COMMUNITY TIP

"I restructured my context files into smaller, focused files instead of one big CLAUDE.md. It improved efficiency by 40-60% for most tasks." Smaller, focused files beat one giant file.

THIS WILL BREAK THINGS

Don't paste your entire brand guidelines into CLAUDE.md. Point to the file instead: "For brand guidelines, see @brand-guide.md." Pasting long documents bloats your context and makes Claude less effective.

MINI CHECKUP

- ☐ I created a CLAUDE.md in my project folder (or ran /init)
- ☐ It includes my brand preferences and project description
- ☐ It's under 150 lines
- ☐ I know to update it when Claude keeps making the same mistake

COPY THIS EXACTLY

"I want to build a one-page website for [your product/event/team]. The page should have: a hero section with a headline and subheadline, a section explaining what this is, a section with three key benefits, a testimonial or quote section, and a call-to-action button. Use a modern, clean design. My brand colors are [your colors]. Before building anything, explain your plan step by step."

Claude will walk you through deployment. For most simple sites, it will suggest Vercel or Netlify. Both are free for basic use. Claude handles the setup.

I LEARNED THIS THE HARD WAY

The first time I used Act mode, Claude helpfully reorganized my entire project folder structure "for clarity." It took an hour to undo. Stick with Code mode until you know what you're doing.

Running multiple sessions

The sidebar in Claude Code Desktop shows your sessions. Each session is an independent workspace. You can have one session building a website and another session fixing a spreadsheet automation. They don't interfere with each other.

Click "+ New session" to start a new one.

SAVE YOURSELF 45 MINUTES

Keep different projects in different folders and different sessions. Don't try to build two unrelated things in the same session. Claude gets confused when the context is mixed.

PART FOUR

Talking to Claude So It Actually Gets It

The difference between a frustrating 45 minutes and a working prototype in 10 isn't the tool. It's how you ask.

- Bad ask vs. good ask (with examples)
- The five rules for good prompts
- The interview-plan-build method
- The prompt template
- How to give feedback mid-project
- The loop problem and how to fix it

PART 4 — TALKING TO CLAUDE SO IT ACTUALLY GETS IT

The single most important skill in this guide

How you ask matters more than what you ask. The same request produces wildly different results depending on how you phrase it. This is true for Cowork and Claude Code.

Think of Claude as a very talented freelancer who just started working with you today. They're smart. They're fast. But they know nothing about your company, your preferences, or your definition of "good." The clearer your brief, the better the work.

Bad ask vs. good ask

The bad version

bad-prompt.txt

```
"Make me a landing page."
```

What kind? For whom? What should it do? What color? Claude will guess. It will guess wrong.

The good version

good-prompt.txt

```
"I need a landing page for our Q2 webinar on AI in marketing.
The audience is marketing directors at mid-size companies.
It should have: a hero with the event title and date,
a section with three speaker bios, a registration form,
and a FAQ section. Use our brand colors: navy (#1B2332)
and teal (#0987A0). Keep it minimal and professional.
Before building anything, explain your plan."
```

Same request. Ten times better output. The extra two minutes of writing save an hour of back-and-forth.

REALITY CHECK

You don't need to learn "prompt engineering." You need to communicate clearly. If you can write a decent brief for a freelancer, you can prompt Claude.

The five rules

1. Start with the outcome, not the method

Say "I need a dashboard that shows our monthly sales by region" not "Build me a React app with a Recharts component." You don't need to know the tools. Claude does.

2. Describe your audience

Who will use this? Your boss? Your customers? Your ops team? Claude makes different design decisions depending on the answer.

3. List what you need, specifically

Vague: "Make it look good." Specific: "Use our brand colors. No more than three sections. Include a call-to-action button above the fold."

4. Say what you DON'T want

Claude is enthusiastic. Left unsupervised, it will add features you didn't ask for. If you want something simple, say so: "Keep this minimal. No extra features beyond what I described."

5. Ask for a plan before execution

Always start with: "Before building anything, explain your plan." Review the plan. Adjust. Then give the green light. This alone prevents most bad outcomes.

WORTH KNOWING

Claude remembers everything in the current session but starts fresh each time you open a new one. That's why your CLAUDE.md file (page 15) is so important. It automatically loads your preferences into every new session so you don't have to repeat yourself.

SAVE YOURSELF 45 MINUTES

Keep a document with prompts that worked well. When you find a phrasing that gets great results, save it. You'll reuse it more than you think.

The interview-plan-build method

Rule #5 says "ask for a plan." The community has refined this into a three-step workflow that works dramatically better than a single prompt. It's called the slowdown, and it looks like this:

Step 1: Describe (let Claude interview you)

Start with this exact prompt: "I want to build [X]. Before you write any code, interview me. Ask me questions about what I need, what it should look like, who will use it, and anything else you need to know."

Claude will ask you 5-8 questions. Answer them. This takes about 3 minutes. Those 3 minutes eliminate most "that's not what I wanted" moments later.

Step 2: Plan (get it on paper)

"Based on our conversation, write a plan. List every step you'll take, in order. Save it as PLAN.md so we can reference it."

Now you have a written plan you can review. Read it. Does anything look wrong? Ask questions. This is dramatically easier than spotting problems after something is already built.

Step 3: Build (one step at a time)

"Execute step 1 of the plan. Stop after completing it so I can review before you move to step 2."

By going step-by-step, you catch problems early. One focused step produces better results than one massive "build the whole thing" prompt, every time.

> ⚠ **GOLDEN RULE**
>
> Never let Claude build without showing you a plan first. The 5 minutes you spend reviewing a plan saves 30 minutes of fixing wrong decisions.

Real example: building a webinar landing page

step-1-describe.txt

```
"I want to build a landing page for our Q2 webinar.
 Before you write any code, interview me. Ask me
 questions about what I need."
```

Claude responds with questions: "What's the webinar topic? Who's the audience? How many speakers? Do you need a registration form?" You answer each one. Then:

step-2-plan.txt

```
"Write a plan for this page. Save it as PLAN.md."
```

Claude saves a plan listing: hero section, speaker bios, agenda, registration form, FAQ. You review it. Then:

step-3-build.txt

```
"Build step 1: the hero section. Stop after that."
```

Why step-by-step matters

When you ask Claude to "build the whole thing," it makes hundreds of decisions without consulting you. Many of those won't match what you wanted. The interview catches requirements. The plan catches the approach. Step-by-step execution catches the details.

SAVE YOURSELF 30 MINUTES

Ask Claude to "interview me about what I need before you start building." This single prompt change eliminates the most common source of wasted rebuilds.

The prompt template

Use this as a starting point for any request:

prompt-template.txt

```
"I need [what you're building].
 The audience is [who will use it].
 The goal is [what it should accomplish].
 It needs to include: [specific features].
 Style: [describe the look and feel].
 Brand colors: [hex codes if you have them].
 Do NOT: [things you want to avoid].
 Before building, explain your plan step by step."
```

YOU DON'T EVEN NEED TO TYPE

I stopped using my keyboard months ago. I use **Wispr Flow** to speak to Claude Code directly — I just talk, and it types. Mac also has built-in dictation (press the microphone key twice). If typing feels like a barrier, remove it. The quality of your prompts comes from clarity of thought, not typing speed.

How to give feedback mid-project

Claude built something. It's close but not right. Here's how to get it there:

INSTEAD OF THIS	SAY THIS
"This isn't right"	"The headline is too small and the button should be blue, not gray"
"Make it better"	"The spacing between sections feels cramped. Add more white space"
"I don't like it"	"The overall style is too playful. Make it more corporate and minimal"
"Fix it"	"The registration link goes to the wrong URL"

Be specific. Point to the exact thing. Describe the exact change. Claude responds to precision the way a dog responds to a thrown ball.

REALITY CHECK

"Try again" is the most useless thing you can tell Claude. It will do the same wrong thing with slightly different words. Interrupt, redirect, reframe.

When Claude is stuck (the loop problem)

Sometimes Claude tries to fix something, fails, tries again slightly differently, fails again, and keeps going in circles. You'll know it's happening when you see the same error three times.

When this happens:

1. Click the **Stop** button (or press Escape).
2. Type: "Stop. The approach you're taking isn't working. Let's try something completely different."
3. Describe the problem from scratch. Don't say "try again." Give it new information.

I LEARNED THIS THE HARD WAY

I once let Claude loop for 20 minutes trying to fix a styling bug. It burned through my usage quota making the same mistake over and over. Now I stop it after the second failed attempt and redirect.

PART FIVE

The Safety Stuff

Five minutes now saves a very bad day later.

- The permission system protects you
- Protect your sensitive files
- Undo button: checkpoints
- The golden rules

PART 5 — THE SAFETY STUFF

Claude Code can create files, delete files, and run programs on your computer. That's what makes it powerful. It's also why you need to understand the guardrails.

The permission system protects you

In Code mode (the default), Claude asks before doing anything destructive. It shows you what it's about to do, and you approve or deny. This is your safety net. Use it.

DO NOT SKIP THIS

If Claude suggests running a command you don't understand, ask: "Explain what this command does in plain English before I approve it." Never approve something you don't understand.

Protect your sensitive files

If your project folder contains files with passwords, API keys, or other sensitive information, you need to tell Claude not to read them. Create a file called **.claudeignore** in your project folder (Claude can create it for you):

COPY THIS EXACTLY

"Create a .claudeignore file in my project folder that blocks access to any file starting with .env, any folder called secrets/, and any file ending in .key"

Undo button: checkpoints

Claude Code saves a snapshot of your files before every change. If something goes wrong, click the rewind button in the toolbar. It restores your files to the state before Claude's last change.

> ⚠ **DO NOT SKIP THIS**
>
> Rewind only undoes file changes on your computer. If Claude did something "out in the world" — like deploying a website or sending an API request — rewind cannot take that back.

The golden rules

1. **Start in Plan mode. Always.**
2. **Use "Allow Once" until you're comfortable.**
3. **Don't put sensitive files** (passwords, keys, credentials) in your project folder.
4. **If Claude suggests running a command you don't understand,** ask: "Explain what this command does in plain English before I approve it."
5. **If anything feels wrong, hit Stop.** Nothing bad happens when you stop.

PROJECTS 01–10

The Playbooks

Real projects, step by step. Each one designed for non-developers.

- 01 — Spreadsheet to Dashboard
- 02 — Marketing Website
- 03 — Monday Morning Report
- 04 — Competitive Intel Tracker
- 05 — Customer FAQ Bot
- 06 — Event Landing Page
- 07 — Team Wiki / Knowledge Base
- 08 — Lead Scoring Prototype
- 09 — Finance / Budget Tracker
- 10 — Email Campaign Builder

PROJECTS 01–10 — THE PLAYBOOKS

Each playbook tells you: who this is for, what you'll end up with, the exact first prompt to send Claude, what to watch for, and how long it takes. Follow them in order or jump to whichever one matches your need.

WORTH KNOWING

Difficulty ratings: ● ○ ○ = Cowork only, no code. ● ● ○ = Claude Code Desktop, straightforward. ● ● ● = Claude Code Desktop, a few moving parts.

PROJECT 01 ●●○

Turn a Spreadsheet into a Real Dashboard

Your boss wants a dashboard. You have the data in Excel. Let's skip the six-week dev request.

DETAIL	INFO
Who	Marketing ops, sales ops, CS leads, anyone reporting metrics
What you get	A live dashboard with charts and filters that opens in any browser. Your team bookmarks it and checks it whenever they want.
Tool	CLAUDE CODE DESKTOP
Time	30–60 minutes including back-and-forth on styling

First prompt (copy this):

COPY THIS EXACTLY

"I have a spreadsheet with our monthly sales data. I want to turn it into a dashboard with: a bar chart showing revenue by month, a line chart showing deal count trend, a dropdown to filter by region, and a summary row at the top showing total revenue, average deal size, and number of deals. I have zero coding experience. Walk me through the plan before building anything."

What happens next

Claude will ask you to share the spreadsheet (drag it in or attach it). It will look at the columns and data types, then propose a plan. Review the plan. Say "Go." Claude builds it. You get a link to preview it on your computer.

WORTH KNOWING

Column names matter. If your spreadsheet has a column called "Rev" and you said "revenue" in your prompt, Claude might get confused. Use the exact column names from your file.

To put it online: Ask Claude: "How do I deploy this so my team can access it?" It will walk you through Vercel (free) or a similar service.

PROJECT 02 ●●○

Build Your Team's Marketing Website

A professional site you control, without waiting for the web team.

DETAIL	INFO
Who	Marketing managers, founders
What you get	Multi-page website with home, about, contact. Looks professional. Loads fast. Works on phones.
Tool	CLAUDE CODE DESKTOP
Time	1–2 hours

First prompt (copy this):

COPY THIS EXACTLY

"I need a professional marketing website for [your company/product]. It should have: a home page with hero section, value props, and CTA button; an about page; a contact page with a working form. Style: clean, modern, mobile-friendly. Our brand colors are [your hex codes]. Before building anything, explain your plan."

What happens next

Claude will propose a framework (usually Next.js or Astro), outline the pages, and describe the layout. Review the plan. Once you approve, Claude builds all the pages, styles them, and gives you a local preview link.

WORTH KNOWING

Claude will suggest Next.js or Astro. Both are fine. Say "pick whichever is simpler."

PROJECT 03 ●○○

Automate the Monday Morning Report

The report your boss reads every Monday, built automatically from your data.

DETAIL	INFO
Who	Anyone reporting weekly metrics
What you get	Formatted executive summary from raw data
Tool	COWORK
Time	15 minutes first run, 2 min after

First prompt (copy this):

COPY THIS EXACTLY

"Here's last week's raw data [attach file]. Turn this into a formatted executive summary with: a one-paragraph overview of key metrics, a table comparing this week vs last week, a bullet list of the top 3 highlights and top 3 concerns. Format it as a Word document I can send to my VP."

What happens next

Cowork reads your data, identifies the key metrics, calculates week-over-week changes, and produces a polished document. The second time, just drop in the new data and say "same format as last week."

SAVE YOURSELF 45 MINUTES

This saves 30–45 minutes every week. Over a year, that's 26–39 hours.

PROJECT 04 ●○○

Competitive Intelligence Tracker

Know what your competitors are doing. Organized, searchable, shareable.

DETAIL	INFO
Who	Product marketing, strategy, sales enablement
What you get	Organized competitive tracker spreadsheet
Tool	COWORK
Time	45 min setup, 15 min monthly

First prompt (copy this):

COPY THIS EXACTLY

"I need a competitive intelligence tracker. Create a spreadsheet with columns for: competitor name, product/feature, pricing, positioning, last updated, and source URL. Start with these competitors: [list 3-5 names]. Then I'll drop in research materials for you to extract and organize."

What happens next

Cowork creates the spreadsheet structure, then waits for your research materials. Drop in pricing pages, feature pages, press releases, or screenshots. Cowork reads them all and fills the tracker.

WORTH KNOWING

Update monthly by dropping in new materials and saying "update the tracker."

PROJECT 05 ●●●

Customer FAQ Bot Trained on Your Docs

A chatbot that answers customer questions using your actual documentation, not made-up answers.

DETAIL	INFO
Who	Customer success, support, product teams
What you get	Chat widget answering from your docs
Tool	CLAUDE CODE DESKTOP
Time	3–5 hours

First prompt (copy this):

COPY THIS EXACTLY

"Build me a customer FAQ chatbot that answers questions using our actual documentation. I'll upload our docs. The bot should: search the docs for relevant answers, cite which document the answer came from, say 'I don't know' when the docs don't cover the question, and offer to connect the user with support. Include a clean chat widget interface."

What happens next

Claude will ask for your documentation files. Upload them all. Claude builds a search layer, a chat interface, and connects them. You test it by asking questions you know the answers to.

REALITY CHECK

This is the most complex project in the guide but the most valuable.

PROJECT 06 ●●○

Event Landing Page with Registration

A page for your next webinar, conference, or product launch. Live in an afternoon.

DETAIL	INFO
Who	Event marketers, demand gen, community managers
What you get	Landing page with registration form
Tool	CLAUDE CODE DESKTOP
Time	1–2 hours

First prompt (copy this):

COPY THIS EXACTLY

"I need a landing page for our upcoming [event type: webinar/conference/product launch]. Details: [event name], [date], [time], [speakers if any]. Include: hero with event name and date, agenda or speaker section, registration form that collects name and email, countdown timer. Style: professional but energetic."

What happens next

Claude builds the page with all sections, a working countdown timer, and a registration form. You preview it, tweak the copy, and deploy.

SAVE YOURSELF 45 MINUTES

After the event, tell Claude: "Turn this into a post-event page with recordings and slides."

PROJECT 07 ●●○

Internal Team Wiki / Knowledge Base

Stop answering the same question in Slack for the fifteenth time.

DETAIL	INFO
Who	Team leads, ops managers, anyone onboarding
What you get	Internal wiki with search and categories
Tool	CLAUDE CODE DESKTOP
Time	2–3 hours

First prompt (copy this):

COPY THIS EXACTLY

"Create an internal team wiki / knowledge base. It should have: a home page with search, category navigation (Onboarding, Processes, Tools, FAQ), individual article pages with a clean reading layout, and a simple way to add new articles. Start by creating the structure and 2-3 sample articles. I'll fill in the real content."

What happens next

Claude builds the site structure with navigation, search, and sample articles. You replace the sample content with real information. Add articles by saying "add a new article about [topic]."

WORTH KNOWING

Claude writes fast once it knows the format.

PROJECT 08 ●●○

Lead Scoring Prototype

Score your leads based on your own criteria. No expensive tool required.

DETAIL	INFO
Who	Sales ops, revenue ops, marketing ops
What you get	Scored and ranked lead list from CSV
Tool	CLAUDE CODE DESKTOP
Time	1–2 hours

First prompt (copy this):

COPY THIS EXACTLY

"Build a lead scoring tool. I want to score leads based on these criteria: company size (1-10 points), industry match (1-10 points), engagement level based on email opens and page visits (1-10 points), budget indicated (1-10 points). The tool should take a CSV of leads and output a scored, ranked list. Use simple rules, not machine learning."

What happens next

Claude builds a scoring engine that reads your CSV, applies the rules you defined, and outputs a ranked list. You can tweak the weights by saying "make industry match worth more."

REALITY CHECK

Start with simple rules. Don't let Claude build a machine learning model. Rule-based scoring is transparent and debuggable.

PROJECT 09 ●●○

Personal Finance / Budget Tracker

Track your spending without giving your bank login to another app.

DETAIL	INFO
Who	Anyone tracking personal spending
What you get	Budget tracker with charts and categories
Tool	CLAUDE CODE DESKTOP
Time	1–2 hours

First prompt (copy this):

COPY THIS EXACTLY

"Build me a personal finance tracker. I'll upload my bank statement as CSV. I want: automatic categorization of transactions (groceries, dining, utilities, etc.), a monthly spending summary with charts, a budget comparison (actual vs target for each category), and a simple dashboard I can check weekly. Everything stays local on my computer."

What happens next

Claude reads your CSV, categorizes each transaction, builds charts, and creates a dashboard. You can adjust categories by saying "merge dining and takeout into one category."

DO NOT SKIP THIS

This runs entirely on your computer. Your financial data never leaves your machine.

PROJECT 10 ●○○

Email Sequence / Campaign Builder

Plan and draft an entire email sequence with your brand voice.

DETAIL	INFO
Who	Email marketers, demand gen, lifecycle marketing
What you get	Complete 5-email nurture sequence
Tool	COWORK
Time	20–30 minutes

First prompt (copy this):

COPY THIS EXACTLY

"I'm planning an email nurture sequence for [product/audience]. Create a 5-email sequence with: Email 1: welcome + value prop, Email 2: educational content / how-to, Email 3: case study / social proof, Email 4: objection handling, Email 5: CTA / offer. Write in a [tone: professional but friendly] voice. Target audience: [describe audience]."

What happens next

Cowork drafts all five emails with subject lines, body copy, and CTAs. You review, tweak the voice, and copy them into your email platform.

SAVE YOURSELF 45 MINUTES

Attach a previous email that performed well and say "Match this tone and style."

PART SIX

When Things Go Wrong

It happens. Here's what to do.

- Claude is making things up (hallucination)
- It keeps making the same mistake
- The preview doesn't load
- You're lost and don't know what happened
- You want to start over
- Context rot: why Claude gets dumber

PART 6 — WHEN THINGS GO WRONG

Claude is making things up

Claude sometimes invents features that don't exist, references files that aren't there, or claims something works when it doesn't. This is called "hallucination." It's not lying. It's pattern-matching gone wrong.

JARGON BUSTER

Hallucination — When an AI confidently states something that isn't true. It's not deliberately lying; it's generating text that sounds right but isn't grounded in reality. Always verify claims that seem surprising.

Fix: If something Claude says doesn't seem right, say: "Are you sure? Show me the actual file/output/evidence." Claude will often correct itself.

It keeps making the same mistake

Claude tries a fix. It doesn't work. It tries again, slightly differently. Still broken. Third attempt. Same result.

COPY THIS EXACTLY

"Stop. The approach of [describe what it's doing] isn't working. Let's try a completely different approach."

Fix: Stop it (click Stop or press Escape). Give it new context or constraints. Don't say "try again."

The preview doesn't load

Claude says the site is running but you see a blank page or error.

Fix:

- Check the address bar. It should say **localhost:something**. If it says something else, ask Claude for the correct URL.
- Ask Claude: "The preview isn't loading. Check for errors and fix them."
- If that doesn't work: "Stop the current server, clear any errors, and restart it from scratch."

You're lost and don't know what happened

The project has files everywhere. You're not sure what's working and what's broken.

COPY THIS EXACTLY

"Explain the current state of this project to me like I'm new here. What files exist, what works, what's broken, and what should we do next?"

Claude will audit the project and give you a status report.

You want to start over

Sometimes the cleanest fix is a fresh start. That's fine. It's not failure. It's software development.

1. Create a new empty folder.
2. Open a new session in Claude Code Desktop pointing to that folder.
3. Paste your original prompt (this is why you saved your good prompts).
4. This time, add any lessons: "Do it the same way but don't use [library that caused problems]. Use [alternative] instead."

I LEARNED THIS THE HARD WAY

I spent two hours trying to fix a project that was fundamentally broken. Starting over with the same prompt took 15 minutes and worked perfectly. Don't fall for the sunk cost fallacy.

REALITY CHECK

Professional developers start over all the time. It's called "spiking" — build a quick version to learn, throw it away, build the real version with what you learned. You're not failing. You're iterating.

Why Claude gets dumber the longer you talk

There's a name for this: **context rot**. It's when Claude gets worse the longer your session goes. It starts forgetting your early instructions, repeats itself, or contradicts what it said ten minutes ago. It's not broken. It's full.

Think of Claude's memory like a whiteboard. Every message you send, every file Claude reads, every command it runs gets written on that whiteboard. Eventually the whiteboard fills up, and Claude starts erasing the oldest stuff to make room for new stuff. Your instructions from the beginning of the session? Erased.

How to recognize it

- Claude starts ignoring rules you set at the beginning.
- It suggests changes you already rejected.
- The quality of its suggestions drops noticeably.
- It "forgets" your brand colors, file structure, or preferences.
- It repeats work it already did.
- It gets confused about which files it already changed.

The fix: three commands every user should know

`/context` checks how full your whiteboard is. It shows a visual breakdown of what's taking up space. Run it periodically. If you're above 70%, it's time to act.

`/compact` squeezes the whiteboard. Claude summarizes everything that's happened so far and throws away the details. You can also focus it: `/compact Focus on the dashboard layout changes` to make sure specific details survive the squeeze.

`/clear` wipes the whiteboard completely. Use this between unrelated tasks. If you just finished building a landing page and now want to work on a spreadsheet, start fresh.

GOLDEN RULE

Context is your most precious resource. Protect it. Start fresh for each new topic. Use /compact when things get slow. Think of every session as having a shelf life.

WORTH KNOWING

Claude's memory (called the "context window") is about 200,000 tokens. That sounds like a lot, but file reads, command outputs, and long conversations fill it fast. A single large file can use 10% of your available space.

The "fresh session" rule

If Claude is still acting confused after compacting, start a completely new session. This is not failure. This is the professional workflow. Your CLAUDE.md file (see page 15) makes sure the new session picks up your preferences automatically. Experienced users start new sessions every 20-30 focused exchanges.

The handoff trick

Before closing a long session, ask Claude: "Write a summary of everything we've done, what's finished, and what still needs to be done. Save it as SESSION-NOTES.md." When you start a new session, tell Claude: "Read SESSION-NOTES.md and continue where we left off." This is how you maintain continuity without context rot.

Using subagents to protect your context

When you need Claude to read a long document or research something, say: "Use a subagent to read the brand-guide.pdf and give me a summary." The subagent (a helper that runs in its own separate space) reads the document and sends back a short summary. Your main session stays clean. Think of it as sending an assistant to the library instead of bringing all the books to your desk.

THE HARD WAY VS THE EASY WAY

Hard way: Keep pushing through a degraded session, getting increasingly frustrated as Claude ignores your instructions. **Easy way:** Start a new session. Your CLAUDE.md brings back all your preferences automatically. Total time lost: 30 seconds.

THIS WILL BREAK THINGS

If Claude starts acting confused or contradicting itself, do NOT keep correcting it in the same session. Every correction you type makes the context problem worse. Start fresh.

MINI CHECKUP

- ☐ I know what context rot is and can spot the symptoms
- ☐ I know how to use /context, /compact, and /clear
- ☐ I write a SESSION-NOTES.md before ending long sessions
- ☐ I start new sessions for new topics instead of reusing old ones

APPENDIX

Quick Reference & Glossary

Everything you need to look up, in one place.

- Glossary of terms
- Copy/paste shortcuts
- Pricing (February 2026)
- Useful links
- When to stop and get a developer
- Connecting your tools (MCP)

APPENDIX — QUICK REFERENCE & GLOSSARY

Glossary of terms used in this guide

TERM	MEANING
API	Application Programming Interface. A way for two programs to talk to each other. You don't need to understand this to use Claude Code.
CLI	Command Line Interface. A text-based way to interact with your computer. Claude Code handles this for you.
Deploy	Putting your project on the internet so other people can see/use it.
Diff view	A display showing what changed in a file. Green = added, red = removed.
Git	A version control system that tracks changes to files. Claude Code uses it behind the scenes.
Hallucination	When AI confidently states something untrue. Not lying — pattern-matching gone wrong.
Localhost	Your own computer, acting as a temporary web server. When you see localhost:3000, the site is running on your machine only.
Sandbox	A protected area where Cowork does its work. Can't affect your real files.
CLAUDE.md	A text file in your project folder that stores your preferences. Claude reads it automatically at the start of every session.
Context rot	When Claude gets worse the longer your session goes because its memory (context window) fills up.
Context window	Claude's working memory. About 200,000 tokens. Files, messages, and commands all use space in it.
MCP	Model Context Protocol. Lets Claude Code talk directly to tools like Notion, Slack, and Figma.
Session	One conversation/workspace with Claude. Starts fresh each time.
Subagent	A helper that runs in its own memory space. Useful for reading large files without filling your main session.
Terminal	A text-based interface for giving commands to your computer. Claude Code interacts with it for you.

Copy/paste on your computer

You'll copy and paste prompts throughout this guide. Here's how:

ACTION	MAC	WINDOWS
Copy	⌘ + C	Ctrl + C
Paste	⌘ + V	Ctrl + V
Select All	⌘ + A	Ctrl + A
Undo	⌘ + Z	Ctrl + Z

Pricing (February 2026)

All paid plans include access to Cowork and Claude Code Desktop.

PLAN	PRICE	BEST FOR
Pro	$20/month	Occasional use, trying things out
Max	$100/month	Daily use, real projects, fewer rate limits
Team	$30/user/month	Organizations, shared billing

Useful links

- **Claude Desktop download:** claude.ai/download
- **Claude documentation:** docs.anthropic.com
- **Vercel (free hosting):** vercel.com
- **Netlify (free hosting):** netlify.com

When to stop and get a developer

This guide helps you build a lot. But some things genuinely need a professional developer:

- **Anything that handles payment processing or credit card data.**
- **Systems that connect to your company's production database.**
- **Healthcare, legal, or financial tools with compliance requirements.**
- **Anything with complex user authentication and permissions** (admin/viewer roles across organizations).
- **Infrastructure that other production systems depend on.**

REALITY CHECK

The skill isn't building everything yourself. It's knowing when you can and when you should ask for help. That judgment is what makes you valuable — not the ability to do every single thing alone.

You've reached the end of the guide.
Now go build something.

Connecting your tools (MCP)

MCP stands for Model Context Protocol. In plain English: it lets Claude Code talk directly to the tools you already use. Notion, HubSpot, Google Sheets, Slack, Figma. Instead of copying data out of HubSpot, pasting it into Claude, and copying Claude's output back, you connect them. Claude reads your data directly and writes back to it. No copy-paste. No switching tabs.

The integrations you probably care about

Each one takes about 60 seconds to set up. Open Claude Code, paste the install command, follow any login prompts that pop up, and you're connected.

TOOL	WHAT IT DOES	INSTALL COMMAND
Notion	Read docs, update pages, manage tasks	claude mcp add notion
HubSpot	Access and manage CRM data	claude mcp add hubspot
Figma	Generate code from your designs	claude mcp add figma
Slack	Send messages, search conversations	claude mcp add slack
Zapier	Connect to 8,000+ apps	claude mcp add zapier
Airtable	Read/write records, manage bases	claude mcp add airtable

WORTH KNOWING

MCP connections persist across sessions. Set them up once and they're always available. Think of it as installing an app on your phone: do it once, use it forever.

Setting up your first connection (Notion example)

1. Open Claude Code in your project.
2. Type: `claude mcp add --transport http notion https://mcp.notion.com/mcp`
3. A browser window pops up. Log in to Notion and click "Approve."
4. Done. Now you can say things like: "Read my Q3 planning doc from Notion and use it to build a dashboard."

Managing your connections

Two commands to remember:

- `claude mcp list` shows everything that's connected.
- `claude mcp remove notion` disconnects a specific tool.

REAL EXAMPLE

"Read our brand guidelines from Notion and build a landing page that follows them." Claude pulled the guidelines directly from Notion, matched the colors and tone, and built the page. No copy-pasting. No missed details.

THIS WILL BREAK THINGS

MCP connects Claude to your real accounts with real data. Don't connect production databases or accounts with sensitive financial data until you're comfortable with how Claude Code works. Start with a test workspace if you can.

MINI CHECKUP

□ I understand what MCP connections do
□ I connected at least one tool I already use
□ I know how to list and remove connections

Ofir's Claude Code Field Manual

A practical guide for non-developers who want to build real things — dashboards, websites, automations — without writing code or waiting for IT.

Ofir's Claude Code Field Manual · February 2026

Compiled by Ofir Bloch