

What changed on 16 September 2026

Cowork is not a separate place any more. Anthropic merged it into Claude chat, so the three surfaces in this chapter are now two. Everything this chapter says about *the work* still stands. Read this page first, then read the chapter with one substitution in mind.

CHAPTER TWO ERRATUM · VERIFIED 18 SEPT 2026 · APPLIES TO “THREE DOORS. WHICH ONE DO YOU OPEN?”

What happened

- Claude went from three modes to two: **Claude chat** and **Claude Code**.
- Cowork's abilities did not go away. They moved into chat, and Claude decides for itself when a request needs that heavier, longer-running treatment instead of asking you to pick a tab.
- Docs and Slides arrived at the same time, and Design now works inside a conversation.
- It is reaching Pro and Max plans first over the following weeks, then Team and Free. Enterprise administrators are told at least 30 days before anything changes for their organisation.

WHY THIS MATTERS MORE THAN A MENU CHANGE

Anthropic's stated reason for merging them is that people could not tell where a task belonged. That is the exact problem this chapter was written to solve. The answer is now partly built into the product – which is good news, and it does not remove the judgement, it moves it.

How to read this chapter now

Where the chapter says	Read it as
“Switch to Cowork”	Stay in chat. Hand Claude the folder and describe the finished thing.
“The Cowork column”	A description of how Claude behaves on project work, not of a separate tab.

“The borrowed room”

Still accurate. Project work runs in a fresh workspace on Anthropic's servers.

“Three doors”

Two doors, three jobs. The jobs are unchanged.

What did not change

- **Claude Code is still its own thing**, and still the only one with your files, your installed software and your logins. Every limit this chapter describes on that boundary still holds.
- **The decision rule survives.** It was never really about tabs. Ask whether the work needs your machine, your credentials and your tools. If it does, that is Claude Code. If it does not, it is chat.
- **Never paste an API key, a password or an MCP link into a chat.** Unchanged, and more relevant now that one surface does more.

THE HONEST LESSON

This is what moving fast looks like. A chapter written in September was overtaken by a product change two days before it shipped. That is the argument for learning the shape of the work rather than the layout of the buttons: the buttons moved, and nothing you learned here was wasted.

Sources • claude.com/blog/cowork-is-now-claude • techcrunch.com/2026/09/16/anthropic-merges-claude-chat-and-cowork-in-one-interface • fortune.com, 16 Sept 2026. Read and verified 18 Sept 2026. If you are reading this months later, check the current docs – that is the habit this whole series is trying to build.



CHAPTER TWO

Three doors. Which one do you open?

Claude, Cowork and Claude Code are not three versions of the same thing, and they are not a beginner, intermediate and advanced ladder. They are three different working arrangements. Once you can tell them apart, you stop asking "can Claude do this?" and start asking the only question that matters: *where does this particular job belong?*

3 SURFACES 1 DECISION RULE VERIFIED **SEPT 2026** CHECK THE DOCS BEFORE QUOTING

The thirty-second version

If you read nothing else on this page, read this table. Everything after it is explanation.

THE THREE SURFACES, SIDE BY SIDE

	CLAUDE	COWORK	CLAUDE CODE
In one line	Thinks <i>with</i> you	Does the work <i>for</i> you	Runs the machinery <i>behind</i> you
The relationship	A very sharp colleague you talk to	An assistant you hand a project to	An operations team with keys to the building
Best at	Answers, drafts, thinking out loud, decisions	Multi-step deliverables built from your own files	Anything repeatable, scheduled, or wired into your systems
Where work happens	In the conversation	In folders you point it at	On your machine and inside your accounts
Keeps working if you close the laptop	No	Yes in the cloud	Yes
Do you need to be technical	No	No	A little, and less every month
Cost	Works on the free plan	Paid plans	Paid plans
Remembers you as	Facts it picks up	The same facts, plus per-project memory	A system of files you own
Reads your CLAUDE.md	No	No not documented	Yes every session
From your phone	The app	Dispatch	Remote Control, iMessage
Start here if you are	New, or thinking	Drowning in documents	Ready to automate the business

THE TRAP TO AVOID

Most coaches pick one, get comfortable, and try to make it do everything. That is how you end up pasting the same spreadsheet into a chat window forty times. The skill is not mastering one door. It is recognising, in about three seconds, which one this job goes through.

Claude: the one you talk to

USE IT FOR • THINKING, DRAFTING, DECIDING

What it actually is

The chat window at claude.ai, the desktop app, or the phone app. You type, it answers. You paste something in, it works on it. Turn by turn, you and it, in a conversation.

This is the surface almost everyone starts on, and a great deal of people never need to leave. Do not let anyone make you feel behind for living here. A coach who is genuinely fluent at conversation with Claude gets more done than one who half-configured a terminal and gave up.

What it is genuinely best at

- **Thinking out loud.** Working through a pricing decision, a difficult client conversation, a positioning problem.
- **Drafting.** Emails, posts, programme descriptions, sales pages, scripts.
- **Making sense of a thing you paste in.** A contract, a spreadsheet of check-ins, a long transcript.
- **Answering, with your own connected accounts as context.** More on connectors below.

Where it runs out

It is turn by turn. It waits for you. If a job takes forty steps, you are personally supervising all forty. That is fine for a decision and miserable for a process. The moment you catch yourself doing the same conversation for the fourth time, you have outgrown this door for that particular job.

Cowork: the one you hand a project to

USE IT FOR • MULTI-STEP WORK ON YOUR OWN FILES

What it actually is

Anthropic describe Cowork as bringing Claude Code's agentic capability to knowledge work beyond coding. In plain terms: you point it at a folder, describe the outcome you want, and it works through the steps on its own until the thing is finished.

It reads, edits and creates files in the folders you give it access to. It can run commands in an isolated environment on Anthropic's servers. It can drive a browser, opening pages, clicking, typing and filling forms. It can run several streams of work in parallel. When it runs in the cloud, the work continues after you close your laptop, and scheduled work can run with no device of yours online at all.

Why this matters to a coach

This is the first door where you stop supervising and start delegating. "Take these eleven consult transcripts, pull out every objection anyone raised, group them, and write me a one-page summary with the three I should build content around" is a Cowork job. In chat it is an afternoon of babysitting. Here it is a sentence and a coffee.

The honest limits

- It is **in beta** on several platforms and the rollout differs by plan. Desktop for macOS is the most established; web and mobile are newer.
- It is **paid only**. There is no free tier.
- It works in **a borrowed room**, not at your desk. More on that below, because it is the single most useful thing to understand about Cowork.

The borrowed room

Same Claude, different desk. Cowork runs in an isolated, temporary environment on Anthropic's servers, separate from your computer. Each session gets a fresh one, and it is deleted when the session ends. It reaches your files only through the Claude Desktop app, and only the folders you have connected.

The part that matters: **it cannot run software installed on your computer, and it cannot use credentials stored on your computer**. Nothing you set up there survives the session, and nothing of yours is there unless you hand it in.

Claude Code is the opposite. It works at your desk: your installed tools, your logins, your saved keys, still sitting there tomorrow. That is the line between them.

A real example: sending a client their intake form

Here is a skill we run for our own coaching business. You say "send Sarah the intake form for the 12-week package," and it creates or updates her contact in the CRM, builds the right link so she

lands on the contract rather than just the waiver, and fires the automation that emails her. Walk it through both doors:

WHY THIS JOB CANNOT RUN IN COWORK			
STEP	WHAT IT NEEDS	COWORK	CLAUDE CODE
Fetch the CRM key	A tool installed on the computer, plus a key file saved on it	No	Yes
Find or create the contact	That key	No	Yes
Build the right link	Just logic: which package, is it a re-sign	Yes	Yes
Fire the email automation	That key again	No	Yes

Cowork could *write* this skill for you. It could not *run* it. The one step it can do is the one that does not matter. Notice it is not the internet that blocks it: the CRM is a public website and Cowork can browse the web. What blocks it is that the key and the tool live on your machine.

So which jobs go where

SORTING REAL COACHING WORK	
COWORK: FILES IN, FINISHED THING OUT	CLAUDE CODE: NEEDS YOUR DESK
Eleven consult transcripts into one objections summary	Sending the intake form: CRM key, contact write, email trigger
Your StoryBrand workbook into a messaging draft	Pulling this week's check-ins out of Trainerize
A transcript plus intake answers into a client one-sheet	Updating your client tracker in Google Sheets
A competitor's website into a comparison doc	Rules that fire every time something happens
A folder of receipts into an expenses summary	Reorganising a year of files and keeping every link working

THE RULE

If the job reaches your stuff through a connector or a folder you hand it, Cowork can do it. If it needs a key you manage or a tool you installed, it is Claude Code. For a Trainerize trainer that splits cleanly: Gmail and Calendar have connectors, so they are Cowork or chat jobs. Trainerize, your intake forms and your Google Sheet client tracker have APIs but no connector, which makes them Claude Code jobs. That is why onboarding, the thing you most want automated, lands on the Claude Code side.

SAY THIS TO A NERVOUS TRAINER

The borrowed room is a benefit, not just a limit. Cowork physically cannot wreck your laptop or leak a password, because it never has them. That makes it the right first step, and Claude Code the move once you have outgrown it.

Claude Code: the one with keys to the building

USE IT FOR • SYSTEMS, SCHEDULES, ANYTHING WITH AN API

What it actually is

Claude running on your own machine, with your files, your terminal, your credentials and your connected systems. It reads and writes anything you let it, runs commands, talks to databases, calls APIs, and remembers how you work through instruction files you write once.

It runs in a terminal, in a desktop app, on the web, and inside code editors. Despite the name, **it is not only for code**. It is the best available tool for operating a business made of files, data and connected software, which is exactly what an online coaching business is.

What only this door gives you

- **Skills** that encode your processes, and **hooks** that fire automatically on events, so a thing happens whether you remember it or not.
- **Subagents**, so heavy reading happens somewhere else and only the answer comes back to you.
- **Scheduled and recurring work** against your real systems.
- **The command line**, which is the part that quietly unlocks everything else. The rest of this chapter is about that.

The honest limits

There is a real setup cost, and it is the only door with one. You install something, you meet a terminal, and for about an evening you feel out of your depth. That is the whole toll. It is smaller than it looks, and it is the last time you pay it.

Remembering: facts versus a system

The most common belief about this is slightly out of date, and the correction matters.

You will hear that chat "forgets everything" and you have to re-explain yourself every time. That was true once. Chat now has **memory**: it saves things as you talk, you can read and edit every one of them in Settings under Memory, and you can simply say "remember this." It is on by

default on the Free, Pro and Max plans, and **chat and Cowork share the same memory**, so what one learns the other knows. Cowork projects add their own project memory on top, scoped so one project does not leak into another.

So chat is not weak at remembering *facts*. What it cannot do is maintain a *system*.

WHAT EACH SURFACE ACTUALLY REMEMBERS		
	CLAUDE AND COWORK	CLAUDE CODE
What it keeps	A list of facts and topics it picked up	Files you own: instructions, memory, skills, client notes, docs
Who decides the structure	Claude	You. The seven drawers in Chapter Three.
Standing instructions	Typed into Settings, or into a project	A <code>CLAUDE.md</code> file, read every session
Your procedures	A skill you create in chat	A skill in a folder, loaded when relevant
Rules that fire automatically	No	Hooks
Undo yesterday's change	No	Yes version history
Updates its own reference files as it works	No	Yes

Does Cowork read your CLAUDE.md?

Not according to Anthropic's documentation. Cowork takes standing context from **global instructions** you type once in Settings, **project instructions** scoped to one project, and the folders or links you point it at. None of Anthropic's Cowork pages mention `CLAUDE.md`. Treat it as no.

That is a bigger difference than it sounds. Your `CLAUDE.md` is a file you own, version and can hand to anyone. Cowork's instructions live inside the app. Improve one and the other does not change.

WHERE "LEARNING AS IT GOES" REALLY LIVES

If you want Claude to *get better at your business every week*, writing down each correction, updating its notes on how Trainerize behaves, sharpening a skill after it fumbles, that is Claude Code. In chat, the best move is to capture a repeated procedure as a skill the moment you notice you are explaining it again. That works, but it is you doing the maintenance. In Claude Code, keeping the files current is part of the job.

Voice: the feature coaches underrate most

Of everything on this page, this is the one that changes a coach's day fastest, and it is the one almost nobody sets up.

Voice mode lets you speak to Claude and listen to it answer. In hands-free mode it listens continuously and replies at the natural pauses in your speech. You can switch between talking and typing inside the same conversation and the context carries over, so nothing is lost when you move from the car to your desk.

It is a beta feature, it is available on **every plan including Free**, and it runs on the mobile apps, desktop and web. It is built to work best from your phone.

THE PART THAT MAKES IT POWERFUL

Voice mode can use web search *and the tools you have connected* during the conversation. That means you can be walking between clients and say "what does my Thursday look like, and did that intake form ever come back from Sarah?" and it can actually go and look. It is not a dictation toy. It is your whole connected setup, operated by talking.

Why this fits coaching specifically

- **Your hands are busy and your brain is not.** Between sets, on the gym floor, driving between sessions, walking the dog after a consult.
- **The ideas arrive at the worst moments.** The programme tweak, the content hook, the thing you meant to tell a client. Voice catches them before they evaporate.
- **Post-consult debriefs.** Five minutes of talking in the car turns into a written summary and a task list by the time you are home.

NOT THE SAME THING AS DICTATION

Dictation just turns your speech into text in the message box, and you still read the reply. Voice mode is a spoken conversation in both directions. Both are useful. Dictation is better when you want a long, precise brief on the record. Voice is better when you want to think with something.

On the Free plan voice gives you Haiku and one connected tool. Paid plans open up the bigger models and all of your connected tools. Voice conversations count against your normal usage limits.

Running it from your phone

This is the part that changes how the work feels. Your computer keeps working at home, and you reach it from the gym floor.

There are three ways to do it, and they are not interchangeable. The simplest way to keep them straight:

- **Remote Control is steering.** You drive one Claude Code session on your computer, live, from your phone.
- **Dispatch is delegating.** You describe an outcome, it splits the work into tasks, runs them, and tells you when they are done.
- **iMessage is texting.** You message your running Claude Code like a person, and the reply comes back in the same thread.

THREE WAYS TO REACH YOUR COMPUTER FROM YOUR PHONE			
	REMOTE CONTROL	DISPATCH	IMESSAGE CHANNEL
Part of	Claude Code	Cowork	Claude Code, as a plugin
You get	The live session, from any device	A manager that runs tasks for you	A chat thread with your session
Your full setup: files, CLAUDE.md, skills, hooks	Yes	For coding tasks. Knowledge tasks go to Cowork, which does not read CLAUDE.md.	Yes
Start new work from the phone	Yes, in server mode	Yes, that is the point	Yes, into the running session
Approvals	Push notification, approve on the phone	Forwarded to you; auto-denied after 10 minutes	Pauses unless the channel relays them
Must stay running	The Claude Code process	Claude Desktop open, computer awake	A Claude Code session with the channel on
If the laptop sleeps	Reconnects when it wakes, messages queued	Nothing runs until it wakes	Messages only arrive while it is running
Plans	All plans; Team and Enterprise need an admin	Pro and Max, limited beta	Pro and Max; Team and Enterprise need an admin
Status	Available now	Beta	Research preview, Mac only

Which one, when

- **You started something at your desk and want to keep going from the couch.** Remote Control. You get the exact session, mid-thought, with everything it already knows.
- **You want to hand off a pile of work and forget about it.** Dispatch. "Summarise this week's check-ins and draft replies for the three who went quiet" is a Dispatch brief.
- **You have a quick question between clients.** iMessage. It is the lowest-friction of the three because you are already in Messages all day.

THE ONE THING THEY ALL NEED

Your computer has to be on. All three do the work on your machine, which is exactly why they can use your files and your keys. A laptop that sleeps in your bag is a laptop that is not working for you. The setups that feel magical are the ones with a desktop or a mini at home that never sleeps.

THE DISPATCH GOTCHA

Dispatch sends coding tasks to Claude Code and everything else to Cowork. Your `CLAUDE.md`, your hooks and your installed tools come along for the first kind and not the second. If a dispatched task ignores a rule you are sure you wrote down, that is almost always why.

The browser, and using your screen

When there is no connector and no API, Claude can use the website or the app the way you do: by looking at it and clicking.

Claude reaches for things in a fixed order, and knowing the order explains almost everything about speed and reliability. **A connector first. The browser second. Your screen last.** Each step down is slower and more fragile than the one above it, so it is used only when the step above is not available.

WHO CAN DRIVE A BROWSER, AND WHO CAN DRIVE YOUR SCREEN			
	CLAUDE CHAT	COWORK	CLAUDE CODE
Search the web	Yes	Yes	Yes
Drive a browser: open, click, fill forms	Through the Chrome extension, which runs as a Cowork session	Yes: a built-in browser, or your Chrome	Yes, through the Chrome extension
Use your logged-in accounts in Chrome	Yes	Yes	Yes
Control your whole screen and your apps	No	Yes, Pro and Max	Yes, Pro and Max

The Chrome extension

Claude in Chrome reads, clicks and navigates websites alongside you, on every paid plan. It opens its own tabs but **shares your browser's login state**, so it can reach any site you are already signed into. That is what makes it useful, and it is also why you should be deliberate about what you let it near. Claude Code can connect to the same extension to test a web page, fill a form or pull data out of a site.

Computer use

The last resort, and sometimes the only option: Claude opens apps, clicks and types on your actual screen. It works in both Cowork and Claude Code, on Pro and Max only, on macOS and Windows through the Claude Desktop app, and your computer has to be awake. It is in beta, and it is honest about its limits: screen control is slower than a connector, and complex tasks sometimes need a second try.

WHERE IT EARNS ITS PLACE FOR A TRAINER

The legacy studio software with no API. The admin screen of a platform that only exists as a website. The report you can export only by clicking through four menus. For those, screen control turns "I have to do this by hand" into "I asked for it." For everything that has a connector or an API, use that instead. It is faster, and it does not break when someone moves a button.

Connectors and CLIs: reach versus depth

This is the section that explains why serious operators end up in Claude Code, and it is not the reason most people assume.

What a connector is

A connector plugs one of your accounts into Claude so it can act there without you copying and pasting. You authorise it once in settings and it works everywhere your Claude account goes, including on your phone and inside voice mode. Google Workspace connectors are available on Claude and Claude Desktop.

They are more capable than people expect. Claude can search, read and analyse Gmail, and it can **draft, send, reply to and forward** email, asking your approval before anything goes out. It can check availability across you and your attendees, **create** calendar invites with Meet links and room booking, **update** events, **delete** them with cancellation notices, and RSVP for you. Drive is read-and-search by default, and you can tighten or loosen each permission in the connector's settings.

CORRECTING A COMMON CLAIM

You will still hear people say the Google connectors are read-only and that you need Claude Code to send an email. That was true once. It is not true now. If your only Google jobs are conversational, the connector is genuinely enough, and it works on your phone, which Claude Code does not.

So where do connectors actually run out?

Not on capability. On **breadth and repetition**. A connector exposes a curated set of actions across a few services. A command-line tool exposes the whole API.

GOOGLE WORKSPACE: THE CONNECTORS VERSUS A CLI, MEASURED		
	GOOGLE CONNECTORS	GWS COMMAND-LINE TOOL
Google services covered	3 Gmail, Calendar, Drive	17 the list below
Sheets	No	Yes
Forms	No	Yes
Contacts	No	Yes
Docs and Slides, writing	No	Yes
Tasks, Chat, Meet, Keep	No	Yes
Depth per service	A curated set of actions	Every method the API exposes
Bulk operations	Not designed for it	Built for it, with auto-pagination
Runs on a schedule, unattended	No	Yes
Works on your phone	Yes	No
Works in voice mode	Yes	No
Setup	Click authorise, once	Install and authenticate

The seventeen services the CLI reaches: drive, sheets, gmail, calendar, docs, slides, forms, tasks, people, chat, meet, keep, classroom, admin-reports, events, modelarmor and cross-service workflow.

Look at that list as a coach. **Sheets, Forms and Contacts are not connector-reachable.** Your client tracker, your intake forms and your actual client list are three of the most important things you own, and they sit outside the curated set.

THE RULE WORTH REMEMBERING

Connectors are for conversation. Command-line tools are for operations. "Did Sarah reply?" is a connector question. "Find every client who has not been emailed in 30 days, cross-reference the training log, and draft the check-ins" is a CLI job. One is a question. The other is a process.

The command line, explained properly

This is the concept that unlocks the rest, and it is taught badly almost everywhere. Here is the version that actually lands.

What it is

Every application you use has a menu of things it can do. Send this. Delete that. List these. When you click a button, you are picking an item off that menu with your mouse.

A **command-line tool** is that same menu, typed instead of clicked. Nothing more mysterious than that.

```
# "list my next 10 calendar events"  
$ gws calendar events list --params '{"maxResults": 10}'
```

That is a menu selection. Service, thing, action, details. Once you can see that shape, the intimidation drops away, because every command you will ever meet is that same four-part sentence.

Why typed beats clicked

Not because it is faster for one action. It usually is not. It wins because of what a typed instruction can do that a click never can:

- **It can be repeated exactly.** A thousand times, identically, with no drift and no fatigue.
- **It can be scheduled.** Run at 6am every Monday without anyone being awake.
- **It can be combined.** The output of one becomes the input of the next, so a chain of tools does a job no single app offers.
- **It can be written down and version-controlled.** A process that lives in a file survives you forgetting it.
- **It can be handed to an AI.** This is the one that matters. Claude cannot click your mouse. It can absolutely type.

THE POINT MOST PEOPLE MISS

You are not learning the command line so you can type commands. You are learning it so that **Claude can**. Your job is to describe the outcome. Its job is to know the syntax. You need to recognise roughly what is happening and when to say stop, which is a far smaller skill than writing it yourself.

Why this means "almost any software you use"

Chapter One defined an **API**: the defined way one piece of software talks to another without a human clicking. Nearly every serious tool you pay for has one, including Trainerize, your CRM, your booking system and your email platform.

An API on its own is not much use to you, because talking to one normally takes code. A **CLI wraps that API in typed commands**. And Claude Code can run typed commands. Chain those two facts together and you get the sentence this whole chapter exists to deliver:

If a platform has an API, it can have a CLI. If it has a CLI, Claude can operate it.

That is the entire unlock. Not a specific integration someone else decided to build for you. Any platform, including the niche one your business depends on that no integration directory has ever heard of.

And if it does not have a CLI, you build one

This is where it stops being a concept and starts being an afternoon's work.

Most platforms a coaching business runs on have an API and no command-line tool. Trainerize has an API. So does Amelia, the WordPress booking plugin. So does Formaloo. None of them shipped a CLI, so none of them were reachable.

So we wrote a **skill** that builds them. You describe the platform once, its base address, how it authenticates, where the credentials live, and it generates a complete working command-line tool in the same shape every time: consistent commands, consistent output formats, credentials pulled from secure storage rather than pasted anywhere.

What that has produced so far

COMMAND-LINE TOOLS BUILT IN-HOUSE FOR PLATFORMS THAT SHIPPED NONE		
TOOL	WHAT IT REACHES	WHY IT MATTERED
trainerize	The Trainerize API	Programmes, clients and messaging, operable by Claude
amelia	WordPress booking	Appointments and availability across every coach site
formaloo	Forms and submissions	Intake data out of a walled garden
mindbody	Studio management	A platform with no useful automation surface
n8n	The automation platform itself	Automations that manage automations
ploi	Server management	Infrastructure without a control panel visit

Six tools, for six platforms that never intended to be operated this way. That is not a large engineering project. It is a repeatable pattern applied six times, and the sixth took an hour.

WHAT THIS MEANS FOR YOU

Stop asking whether a platform "integrates with AI". Ask whether it has an API. If it does, it is reachable, and the gap between reachable and reached is much smaller than it used to be.

Which door, in three seconds

The decision rule

- 1. Am I thinking, deciding or drafting?** That is Claude. Use voice if your hands are busy.
- 2. Is there a pile of my own material that needs turning into a finished thing?** That is Cowork.
- 3. Will I need this to happen again, on a schedule, or against a system nobody built an integration for?** That is Claude Code.

Almost every job you have sorts cleanly into one of those three. When a job seems to want two doors, it is usually two jobs.

A worked week

- **Monday, driving:** voice, on the phone. "What is on today and did anyone cancel?"
- **Monday, at the desk:** Claude. Think through a pricing change for the new programme.
- **Tuesday:** Cowork. Eleven consult recordings into one objections summary.
- **Wednesday:** Claude Code. Build the check-in chase so it runs itself every Monday from now on.
- **Thursday, on the gym floor:** a text to your Claude Code over iMessage. "Send Marcus the intake form for the 12-week." It runs at home, on your machine, with your keys.
- **Friday, before bed:** Dispatch. "Summarise this week's check-ins and draft replies for anyone who went quiet." It is waiting for you in the morning.
- **From then on:** Wednesday's work happens without you. That is the whole point.

WHERE TO START IF THIS IS ALL NEW

Turn on voice mode and one connector this week. Do not touch a terminal yet. Get fluent at the conversation first, notice which jobs you keep repeating, and let that list tell you when it is time for the third door. The list will build itself.

Capabilities here were verified against Anthropic's documentation in September 2026. These products move quickly and several are in beta, so check the current docs before betting a business process on any single row.

FITNESS MARKETING MACHINE CONCORD, NC FITNESSMARKETINGMACHINE.NET
SCALE WITHOUT LIMITS