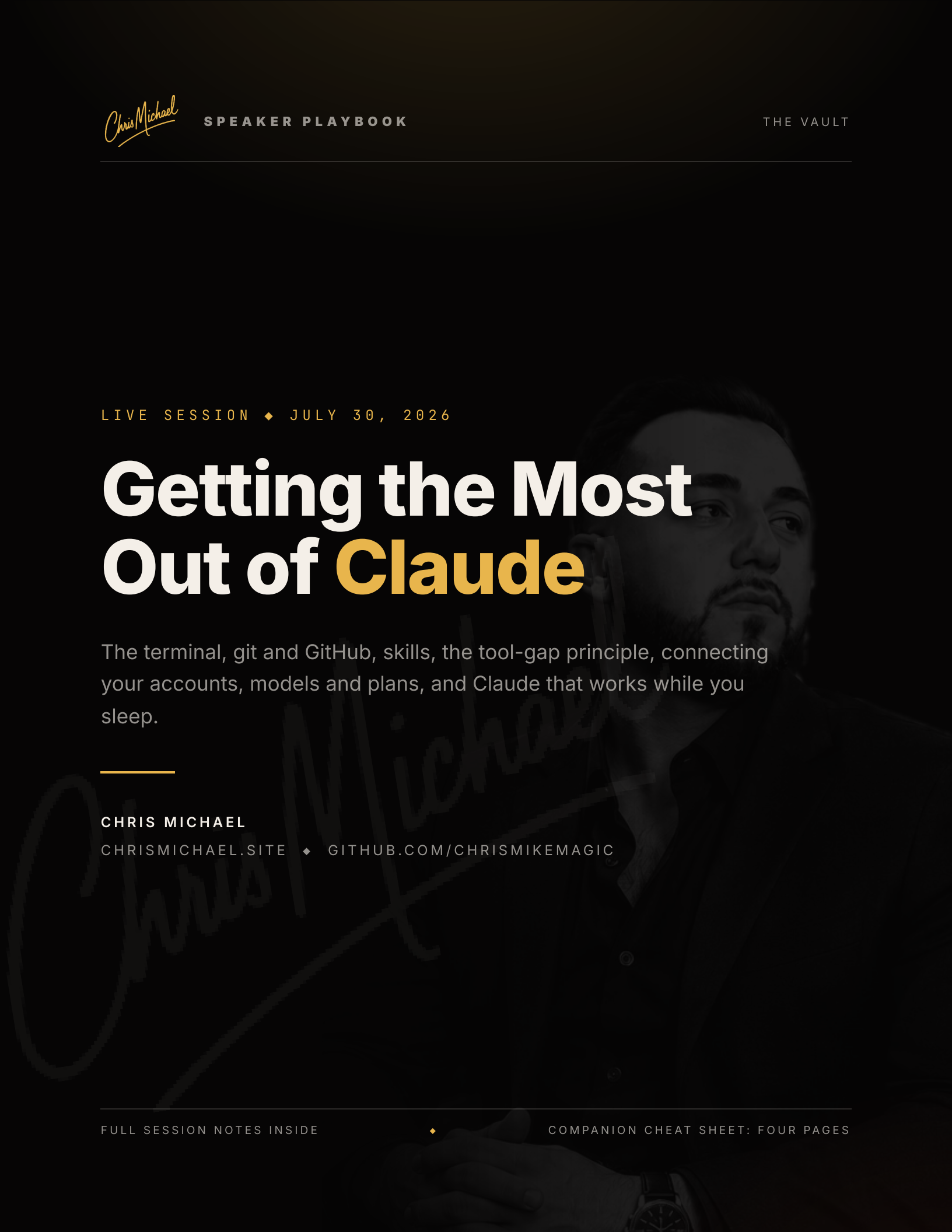
LIVE SESSION ◆ JULY 30, 2026

Getting the Most Out of Claude

The terminal, git and GitHub, skills, the tool-gap principle, connecting your accounts, models and plans, and Claude that works while you sleep.

CHRIS MICHAEL

CHRISMICHAEL.SITE ◆ GITHUB.COM/CHRISMIKEMAGIC



Speaker Playbook

Everything you are teaching, in the order you are teaching it.

TERMINAL · GIT · GITHUB · SKILLS · TOOL GAP · YOUR ACCOUNTS · EMAIL · DASHBOARDS · MODELS · ALWAYS ON · DEMOS · TOOLKIT · PROMPTS · **THE CLOSE**

01	**Claude in the terminal**	install, log in, who can use it
02	**Why use the terminal**	the bicycle and the motorcycle
2b	**Living in the terminal**	restarts, handoffs, dragging files
03	**What git is**	the easy explanation
04	**Why GitHub matters**	backup, sharing, the hub
05	**How to add skills**	recipe cards for Claude
06	**Claude can do almost anything**	the tool-gap principle
07	**The four doors into your accounts**	the four doors
08	**Getting Claude to send email**	what everyone gets stuck on
09	**A dashboard to control Claude**	status files and buttons
10	**Models, effort, ultracode, plans**	and why $20 will not do it
11	**Claude while your computer is off**	cloud, routines, your phone
12	**Demos**	Netlify, GitHub, iMessage
13	**My Claude toolkit**	four open tools
14	**Prompts to hand the room**	five, word for word
15	**The close**	the offer, last two minutes
16	**Prep checklist and likely questions**	run this before you start

01 Claude in the Terminal

CLAUDE CODE

The pitch: the Claude you know from the chat app answers questions. Claude Code **does** things. It lives on your computer, can read and write your files, run programs, and use the internet. Same brain, but with hands.

OPENING THE TERMINAL (DO THIS ON SCREEN)

- **Mac:** press Cmd+Space, type "Terminal", hit Return
- **Windows:** press the Windows key, type "PowerShell", hit Enter

> **WHY IT MATTERS**
>
> For half the room this is the scariest step in the whole session, and it takes four seconds. Doing it live defuses it.

INSTALLING (ONE LINE, THAT IS THE WHOLE THING)

- Mac: `curl -fsSL https://claude.ai/install.sh | bash`
- Windows: `irm https://claude.ai/install.ps1 | iex`
- Nothing else needed first. No developer tools, no Node, nothing
- **Mac, do not skip this:** when the installer finishes it prints a line starting with `echo`. Copy that whole line, paste it back into the terminal, hit Return. That is what makes the `claude` command work
- Then type `claude` in any folder. First run opens a browser to log in with your normal Claude account

> **FIRST THING TO DO ONCE YOU'RE IN**
>
> Run prompt P1 from section 14, the one-letter launch, so that typing `c` starts Claude with permission prompts skipped. Set it up before anything else and the rest of the session moves twice as fast.

WHO CAN USE IT

Any paid Claude plan: Pro, Max, Team, or Enterprise. Not the free plan.

WHERE ELSE IT LIVES

A desktop app for Mac and Windows, a web version at claude.ai/code that runs in Anthropic's cloud (start a job, close your laptop, check it from your phone), and IDE extensions. For beginners, terminal or desktop app is the move.

SAY THIS

You don't need to know terminal commands. You open the terminal to start Claude, then you just talk to it. Claude types the commands.

02 Why Use the Terminal

THE BICYCLE AND THE MOTORCYCLE · GIVE IT TWO MINUTES

THE METAPHOR, WORD FOR WORD

The Claude app is a bicycle. The terminal is a motorcycle. They both get you where you want to go, but the bicycle takes much longer and it's your legs doing the work. The motorcycle is faster and more powerful, and yes, that comes with more risk, but the risk is managed for you and it's nothing to worry about at our level.

THEN THE ARMS LINE

The app is a brilliant talking head. It can tell you exactly what to do, and then YOU go do all of it. The terminal gives Claude arms. Same brain, but now it does the work.

WHAT THE ARMS CAN DO (RATTLE THESE OFF)

- Search and read every file on your computer
- See what's on your desktop and organize the whole machine
- Edit videos, resize photos, convert anything into anything
- Diagnose and fix what's broken. You have a permanent IT partner
- Build and ship real things: websites, documents, apps, automations that run while you sleep

THE LINE THAT LANDS

In the app you receive advice. In the terminal you receive finished work.

OWNERSHIP

"In the app, your work lives inside a chat window. In the terminal, everything Claude produces is a real file on your computer. The chat can end; the work stays. Tomorrow you can switch models, bring in a different AI, even run one locally someday, and everything that was built is still sitting there, yours forever, no login required."

COMPOUNDING

"A chat forgets. Your computer doesn't. In the terminal, Claude writes what it learns into files that persist: your preferences, your projects, your history, your standards. Every session starts smarter than the last one, because the last one left notes. The app gives you conversations. The terminal builds you an asset that compounds."

RISK, ADDRESS IT HEAD ON

"More power means more responsibility, the way a motorcycle asks more respect than a bicycle. But Claude asks permission before touching anything that matters, and git puts a save point behind every change, so nothing is ever truly lost. This is a solved problem. Ride the motorcycle."

CLOSER

Bicycles are for rides around the block. Motorcycles are for actually going somewhere.

2b Living in the Terminal

THE EVERYDAY HABITS NOBODY TELLS YOU

NEW TOOLS NEED A FRESH START

Whenever you install a skill, a plugin, or an MCP connection, the safe move is to **quit Claude and open a new one**. Claude reads its available tools when it starts up, so anything added mid-conversation may not exist as far as it knows.

- Simple skills are often picked up live, but plugins and MCP connections need a restart to load
- You do not need to close the terminal window itself. Quitting Claude and typing `c` again is enough
- The tell: you ask for the new thing and Claude says it can't. Restart before you troubleshoot anything else

> **THE TRAP**
>
> People install something, watch it not work, and conclude the tool is broken. Nine times out of ten they simply never restarted.

PICKING UP WHERE YOU LEFT OFF

COMMAND	WHAT IT DOES
claude --continue	Reopens your most recent conversation in this folder, full history intact
claude -c	The short version of the same thing
claude --resume	Shows a list of past conversations so you can pick one

These only find conversations started **in the same folder**, which is one more reason to keep each project in its own folder.

THE HANDOFF PROMPT (TEACH THIS ONE, IT IS THE PRO MOVE)

Restarting does not have to mean losing your place. Before you quit, ask for a handoff.

```
Write me a prompt I can paste into a fresh Claude session to continue exactly this work. Include what we're building, what you've done so far, what's left, and any decisions I've made that you should keep.
```

- You get a paragraph you paste into the new session, and it picks up mid-stride
- This also works across machines: write the handoff on your laptop, paste it on your desktop
- Ask for it before installing anything, before a long break, or any time a conversation gets huge and slow

GETTING FILES AND IMAGES INTO CLAUDE

- **Drag any file from Finder or File Explorer onto the terminal window.** It types the full path for you, no typing or spelling
- Claude reads that path and opens the file, whether it's a photo, a PDF, a spreadsheet, or a video
- **Mac:** you can also copy a file's path in Finder with Cmd+Option+C, then paste it in
- **Windows:** you generally cannot paste an image itself into the terminal, but you can give Claude the location. In File Explorer press **Ctrl+Shift+C** to copy the file path, then paste that. Shift plus right-click, then "Copy as path," does the same thing

> SAY THIS
>
> Claude does not need you to attach anything. It needs to know where the thing lives. Drag the file in and it knows.

03 What Git Is

THE EASY EXPLANATION

SAY THIS

Git is a save system for an entire folder. Every time you save (that's called a commit), it takes a snapshot of every file in the project. You can jump back to any snapshot, ever. It's Track Changes, but for a whole project instead of one document.

BACKUP ANALOGIES IF FACES GO BLANK

- Video game save points. You save before the boss fight. If it goes wrong, you reload
- A photo album of your project. Each page is what the project looked like on a given day

WHY NON-PROGRAMMERS SHOULD CARE

- Claude makes lots of changes fast. Git means nothing Claude does is ever scary, because you can always go back
- You never have final_v2_REAL_final.zip again. One folder, full history

KEY VOCAB, KEEP IT TO THREE

repo	a folder git is watching
commit	one save point
push	upload your saves to the internet

SAY THIS

You will never type a git command. Claude does the git. You just need to know the save points exist, so you're never afraid to let Claude try things.

04 Why GitHub Matters

SEPARATE FROM GIT

SAY THIS

Git is the save system on your computer. GitHub is where those saves live online. Git is the camera, GitHub is the photo album in the cloud.

THREE REASONS IT MATTERS

1. **Backup.** Laptop dies, project doesn't.
2. **Sharing.** Send someone a link and they have your entire project and its whole history. This is how I share my toolkit with you today.
3. **It's the socket everything else plugs into.** This is the big one. Netlify watches my GitHub and republishes my website every time Claude pushes. TestFlight builds, automations, all of it hangs off GitHub. GitHub is the hub; everything else subscribes to it.

BONUS

Claude Code talks to GitHub natively (the `gh` tool), so Claude can create repos, open pull requests, and read other people's projects for you.

05 How to Add Skills

RECIPE CARDS FOR CLAUDE

SAY THIS

A skill is a recipe card you hand Claude. It's literally a folder with a text file in it called SKILL.md that says: when the user asks for X, here's exactly how to do it, step by step, with my preferences baked in.

INSTALLING ONE SOMEBODY SHARES WITH YOU

- Drop the skill's folder into `~/.claude/skills/`, the skills folder inside the hidden .claude folder in your home folder
- Or easier: tell Claude "install this skill from this GitHub link" and it does it
- Start a fresh session and it's live. Invoke by asking naturally, or by typing /skill-name

TWO FLAVORS

KIND	WHERE IT LIVES	BEHAVIOR
Personal	`~/.claude/skills/`	Follows you everywhere
Project	`.claude/skills/` in a repo	Travels with the project, everyone who clones it gets them

Plugins: bundles of skills plus connections, installed with the `/plugin` command. There's an official marketplace. Think "app store for Claude abilities."

SAY THIS

Skills are how you stop re-explaining yourself. Anything you've told Claude twice should become a skill.

06 Claude Can Do Almost Anything

THE TOOL-GAP PRINCIPLE

THE CORE IDEA

When Claude says it can't do something, that's almost never an intelligence problem. It's a tool problem. It means Claude doesn't currently have a way to touch that thing. Your job isn't to accept the no. Your job is to ask: what tool would make this possible?

THE VIDEO EXAMPLE (USE THIS ONE, IT LANDS)

- Ask chat Claude to watch a YouTube video: it can't. It has no eyes on video
- Ask Claude Code the same thing: it installs a downloader (yt-dlp), pulls the video, extracts frames as images with ffmpeg, looks at the frames, and transcribes the audio. It has effectively watched the video, and can summarize it, pull quotes, find the moment something happens
- **Nothing about Claude changed. The tools changed.**

OTHER "CAN'T" BECOMING "CAN"

Reading a phone screen (mirror it), checking a website visually (headless browser), sending a text (Messages access), generating video (a connected generation service).

> THE HABIT TO TEACH
>
> When Claude declines, ask it: "what tool or access would you need to do this?" It will usually tell you, and often it can install that tool itself.

07 The Four Doors Into Your Apps, Sites, and Accounts

THE FOUR DOORS

Frame: "So far Claude can touch your files. The next level is letting it touch your accounts: your website host, your email, your ads, your messages. There are four doors in."

DOOR 1 · MCP (MODEL CONTEXT PROTOCOL)

- Say: "MCP is a standard plug. Companies publish an MCP for their product, you connect it once, and Claude gets a set of buttons for that product."
- Added with `/mcp` or `claude mcp add`, usually a one-time login in the browser
- My live examples: browser control, image and video generation, Meta ads, Canva, documentation lookup

DOOR 2 · APIS AND ACCESS TOKENS

- Say: "For anything without an MCP, most services have an API, and a token is a special password you generate for it. You give Claude the token, and Claude writes and runs the code to work that service directly."
- State the best practice out loud: smallest permissions that work, revoke any time without changing your real password, keep them out of any code you publish
- **Tip:** add an OpenAI API key (platform.openai.com/api-keys) and Claude gains image generation. It writes the code that calls the image model; you get the pictures

DOOR 3 · BROWSER CONTROL

- Say: "For anything with no MCP and no API, there's the everything door: Claude drives a real browser. The Claude in Chrome extension lets it click, type, fill forms, and read pages in your actual Chrome, using your logged-in sessions. If you can do it in a browser, Claude can."
- There's also headless browsing, where Claude runs an invisible browser for checks and testing

> **GIVE THEM THE EXACT STEPS**
>
> Get it at **claude.ai/chrome**. Install it, then OPEN the extension, log in, click Authorize, then scroll to the bottom and set it to take action without asking. People miss that last step and then wonder why it stops for approval on every single click.

DOOR 4 · DEDICATED ACCOUNTS FOR CLAUDE

- Say: "For ongoing jobs I sometimes make Claude its own account: its own GitHub identity, its own login for a service. Then its actions are labeled, its access is scoped, and I can shut it off without touching my accounts."
- Caveat: check the service's terms on automation, and never give a bot account more access than the job needs

08 Getting Claude to Send Email

THE THING EVERYONE STRUGGLES WITH

SAY IT STRAIGHT

The official Gmail connector is deliberately conservative. It can read your mail and write drafts, but it will not hit send. That's a safety choice by Anthropic, not a capability limit. If you've tried and concluded Claude can't send email, you hit a guardrail, not a wall.

THE THREE REAL ROUTES TO ACTUAL SENDING

1. **Draft-first workflow.** Start here. Claude reads the thread, writes the reply as a draft in your Gmail, you tap send. Zero risk, and honestly right for client email until you trust the voice.
2. **Your own access token.** Set up Gmail API access (or an app password with SMTP) and give Claude Code the token. Now it sends for real, from code it writes. Same door 2 from the last section.
3. **Mac shortcut.** Claude can drive Apple Mail with AppleScript, no API setup at all.

THE PATTERN THAT MAKES AUTO-REPLY SAFE

- **Step 1:** Claude drafts, you send. Do this for two weeks
- **Step 2:** Claude sends automatically for one narrow category only (booking inquiries, say), everything else still draft-only
- **Step 3:** widen the categories as trust builds. Always keep a category list, never "reply to everything"
- **Voice matters:** give Claude real examples of emails you actually sent, so replies sound like you and not like AI. This is a skill: your email voice as a recipe card

PROOF IT WORKS

I run a Facebook DM auto-replier that answers around the clock, and iMessage relays that hold real conversations in my voice. Email is the same pattern with a different pipe.

09 A Dashboard to Control Claude

STATUS FILES AND BUTTONS

The question behind the question: people want to glance at one screen, see what Claude is doing, and press buttons instead of typing prompts.

THE TRICK THAT MAKES IT SIMPLE

A dashboard is just an HTML page that reads little status files, and your Claude agents are told to write those status files as they work. The page shows state; buttons run saved prompts. Claude will build the entire thing for you in one sitting, because it's just a web page plus a few scripts.

THE PIECES

- **Status out:** each running agent or scheduled job appends a line to a status file. The dashboard reads and displays those. Heartbeats tell you a job is alive
- **Buttons in:** each button triggers a script that launches Claude with a saved prompt (`claude -p "do the morning routine"`). One click, no typing
- **Where it lives:** a local HTML file, or inside Obsidian if you already live there

MY LIVE EXAMPLE

Mission Control: an Obsidian dashboard over my Claude memory wiki with Command Deck buttons, a machine-written agent status table, a morning brief that writes itself, and heartbeat monitoring. Every part of it was built by asking Claude to build it.

EASIEST FIRST STEP FOR THE ROOM

"Ask Claude Code: build me a one-page HTML dashboard that shows the status of my projects and has three buttons for my three most common requests." That prompt, roughly as written, works.

10 Models, Effort, Ultracode, Plans

AND WHY $20 A MONTH WILL NOT RUN A BUSINESS

THE MODEL LADDER (SWITCH WITH /MODEL)

MODEL	REACH FOR IT WHEN	EVERYDAY EFFORT	CRANK IT TO
Haiku	Chores: renames, bulk edits, lookups	low	medium when output still matters
Sonnet	The daily driver for most building	medium	high for real work sessions
Opus	Hard debugging, big features	high	xhigh for architecture, thorny bugs
Fable 5	Long autonomous jobs, deepest reasoning	high	max for the one problem nothing solved

Nice trick: opusplan mode uses the smart model to make the plan and the cheaper model to execute it. Plan expensive, execute cheap. **Fast mode** (`/fast`): same Opus intelligence, much faster output, about double the cost.

> THE ONE HEURISTIC
>
> Match the model to the stakes and the effort to the difficulty. Effort is how long Claude thinks per step; the model is the ceiling. When output disappoints, raise one of the two dials before you blame Claude.

ULTRACODE, THE SWARM WORD

- Say: "There's one more gear above all of this. Put the word **ultracode** in your prompt and Claude stops being one worker and becomes a fleet: it plans the job, fans out parallel subagents (searchers, builders, and skeptics that double-check every finding), then merges the results into one answer."
- Use it for audits, deep research, and "be exhaustive" jobs, anything where you'd rather have twenty perspectives than one
- Sibling: `/code-review ultra` runs a multi-agent review of a whole branch or pull request

HONEST CAVEAT

On Pro this will eat your cap fast. Swarm mode is a Max-plan habit.

WHY $20 A MONTH LIKELY IS NOT ENOUGH

SAY THIS

The Pro plan is priced for chatting. Agentic work is a different animal. When Claude works on a task, every single step re-sends the whole conversation. One serious working session can chew through more usage than a month of chatting.

PLAN	REALITY
Pro $20	Try it, learn on it. A heavy session can hit the cap mid-task
Max $100	About 5x the usage. The realistic floor for daily business use
Max $200	About 20x. Agents all day: relays, scheduled jobs, multiple projects

Framing so it doesn't sound like an upsell: "Try Pro first. The day Claude stops mid-task and tells you to come back in three hours, and that costs you more than $80 of your time that month, the math has made the decision for you. I'm on the top tier because this runs my whole business."

ACCURACY NOTE

Caps reset on roughly 5-hour windows with weekly ceilings on top, and Anthropic tunes them. Teach the shape, not specific hour numbers.

11 Claude While Your Computer Is Off

CLOUD, ROUTINES, YOUR PHONE

Frame: "Everything so far assumed you're at your desk with a terminal open. You don't have to be. There are three ways to make Claude always-on."

WAY 1 · CLOUD SESSIONS

- claude.ai/code runs sessions on Anthropic's computers, not yours. Start a task, close the laptop, it keeps working
- `claude --cloud "do the thing"` sends a task up; `/teleport` pulls a cloud session down into your terminal to continue locally
- Cloud sessions pull your project from GitHub. Yet another reason GitHub is the hub, callback to section 4

WAY 2 · ROUTINES (SCHEDULED AND TRIGGERED)

- Recurring Claude jobs that run themselves in the cloud: every morning, every hour, weekdays only, or fired by a trigger like a webhook or a new pull request
- Set up with `/schedule` in the terminal, or on the claude.ai/code site
- This is the answer to "I want Claude to check my inbox every morning and draft replies before I wake up" with no computer on

WAY 3 · A MAC THAT STAYS ON (THE DIY VERSION)

- My auto-replier and relays run as background jobs on a Mac that never sleeps. Anything you can script, macOS can run on a schedule or on an event
- Honest comparison: more control and no cloud limits, but you're the one keeping a machine alive. Cloud routines are the beginner path; the always-on Mac is the power path

TWO COMMANDS WORTH DEMOING HERE

COMMAND	WHAT IT DOES
/remote-control	Turns on remote control for the session you are in, so you can steer it from your phone or browser while your machine keeps running. Also `claude remote-control` to start one, which shows a QR code
/loop	Repeats a task on a schedule. `/loop 5m check the deploy` runs it every five minutes. Leave the interval out and Claude paces itself. Ctrl+C stops it

> **ASK BEFORE SOMEONE ELSE DOES**
>
> There is no voice command in the terminal. Voice input lives in the Claude mobile and desktop apps, not the CLI. On a Mac you can still dictate into the terminal with the system dictation shortcut.

FROM YOUR PHONE

- **Official:** the Claude mobile app starts and monitors cloud sessions and pushes a notification when a task finishes or needs approval. For cloud work the phone is genuinely enough
- **Real terminal:** install Tailscale on the Mac and phone (free, makes them a private network), use an SSH app like Termius or Blink, and run Claude inside tmux so the session survives disconnects. You reattach from your phone mid-conversation
- Keep it honest: the Tailscale route is a setup project, not a tap. Offer it as "ask Claude to walk you through setting this up," which is on-message for section 6

12 Demos

NETLIFY, GITHUB, IMESSAGE

DEMO A · GITHUB PLUS NETLIFY (THE MONEY DEMO)

- Ask Claude to make a small change to a site, a copy tweak or a color change
- Claude edits, commits, pushes to GitHub
- Show the Netlify dashboard picking up the push and deploying, then refresh the live site

> **THE LINE**
>
> I never touched a deploy button. Push to GitHub IS publishing. Claude ships my websites end to end.

DEMO B · GITHUB AS SHARING

- Open github.com/chrismikemagic/claude-toolkit in the browser. This doubles as the setup for section 13
- Show the commit history on any repo: "every save point Claude ever made, with a description of what changed"

DEMO C · IMESSAGE ON MAC

- What's happening underneath, in one line: "On a Mac, your Messages history is a database file on disk. Give Claude permission to read it and it can read texts; the Messages app lets it send them. A plugin turns that into a loop: message arrives, Claude reads it, Claude replies."
- What I actually run: an iMessage plugin with an allowlist, so Claude only ever sees chats I've explicitly approved. Mention the auto-replier and the relays as what this unlocks

> **DEMO SAFETY**
>
> Text yourself or a designated test contact. Do not put your real inbox on screen. Close notification-heavy apps and anything with client names first.

13 My Claude Toolkit

GITHUB.COM/CHRISMIKEMAGIC/CLAUDE-TOOLKIT

Frame: "Four open tools I built, all free, MIT licensed. Each solves a different part of one problem: keep Claude's context lean, keep the request sharp, and keep what Claude learns from evaporating between sessions."

01 · CONTEXT-FORGE — SHARPENS THE REQUEST GOING IN

You talk to Claude the way you actually talk: rambling, half-formed. This turns that into an engineered brief (clear goal, constraints, plan, verification steps) before any work starts.

Why it matters: most bad Claude output is really a bad request. Fix the input and the output fixes itself.

02 · BETTER-OPUS-4.8 — STRUCTURES THE WORK

A planning, review, and QA loop: premise checks in fresh context, multiple judged approaches, small commits behind an evidence gate, parallel review lenses.

Why it matters: you get top-tier output through structure instead of paying for it. The scaffolding does the judgment the fanciest models do in their head.

03 · MEMORY-PALACE — KEEPS WHAT CLAUDE LEARNS

Turns Claude's memory from one flat file that rots into a personal wiki it maintains itself: entity pages, links, an Obsidian vault you can browse, a librarian agent, a weekly cleanup pass.

Why it matters: Claude actually knows your business next week. My entire operation runs on mine.

04 · CLAUDE-SETUP-OPTIMIZER — TRIMS SETUP, MAKES HISTORY SEARCHABLE

One prompt that audits what your setup costs in context every session, prunes what you never use with evidence from your real transcripts, then builds a local search index over every past session, wired in so relevant past work resurfaces automatically. One-command rollback.

Why it matters: everything you install rides along in every conversation and dulls the model. This pays for lean, then makes lean cost nothing.

HOW THEY FIT

context-forge sharpens what goes in, better-opus structures the work, memory-palace keeps what matters on purpose, and setup-optimizer makes everything you ever did findable underneath.

14 Prompts to Hand the Room

FIVE, WORD FOR WORD · ALSO ON THEIR CHEAT SHEET

Frame: "I'm going to give you five prompts to paste in word for word. Each one permanently changes how Claude works for you."

P1 The one-letter launch

```
Set up a shell alias so that when I type c in the terminal, it launches Claude with permission prompts skipped. Make it permanent, confirm it works, and briefly tell me what to be careful about when running this way.
```

Coaching note: say out loud that skipping permissions trades safety for speed. Use it in your own projects, not in folders you can't afford to break.

P2 Study my business, then run with it

```
Find the Booked Solid folder on my computer and unzip anything that needs unzipping. Study everything inside it: every system, process, procedure, piece of knowledge, and tool. Save what you learn to your memory so it applies to all of my future Claude projects. From now on, when I ask you to complete a task, do your best to finish it on your own without my input: find or build the tools, MCPs, and skills you need, do your own research, and only come back to me when a decision is genuinely mine to make.
```

Coaching note: this is the delegation switch. One paste turns Claude from an assistant that asks into an operator that finishes.

P3 Your own mission control

```
Build me a personal mission-control dashboard as a beautifully designed single-page website. Before designing anything, study my brand: look at my website, my logos, my photos, and any brand files on this computer, and match my colors, fonts, and overall aesthetic. The dashboard should show my projects and what you're currently working on, stay current by reading simple status files that you keep updated, and have buttons for my most common requests. Host it so I can open it any time, and tell me the one link to check each morning.
```

Coaching note: pair it with Obsidian as the knowledge base behind it, free at obsidian.md/download.

P4 The plain-English switch

```
I am not very technical. From now on, explain ideas and instructions in plain language that a non-technical person can understand: short steps, no jargon, and when a technical term is unavoidable, define it in one simple sentence. Before doing anything complicated, tell me what you're about to do and why in a sentence or two. Remember this as a permanent preference.
```

P5 The key vault

```
From now on, any time I share a password, authorization token, personal access token, or API key with you, store it in my computer's keychain so it is never lost and always accessible in a secure way. When you need it later, read it from the keychain instead of asking me again. Never write secrets into plain-text files, into code, or into anything that gets pushed to GitHub. Remember this as a permanent rule.
```

Coaching note: this is how my own setup runs. Keys live in the Mac Keychain, encrypted, and Claude pulls them when needed. On Windows the same idea uses Credential Manager.

EVERY LINK TO GIVE THE ROOM

- **Claude Code docs:** code.claude.com/docs · **Cloud sessions:** claude.ai/code
- **Browser extension:** claude.ai/chrome (install, open it, log in, Authorize, then set take action without asking)
- **Connect apps (MCP):** claude.ai/settings/connectors, or `/mcp` in the terminal
- **API keys:** OpenAI platform.openai.com/api-keys · Anthropic platform.claude.com · GitHub github.com/settings/tokens · Google console.cloud.google.com
- **Ship websites:** github.com + netlify.com · **Phone terminal:** tailscale.com + termius.com
- **Knowledge base:** obsidian.md/download · **My toolkit:** github.com/chrismikemagic/claude-toolkit

15 The Close

THE OFFER · LAST TWO MINUTES

THE TURN

Everything I gave you today is tools, and tools are the easy part. What decides who wins a category is behavior. That is the work I actually do with companies.

THE DISTINCTION TO DRAW

Today was: how to install it, how to connect it, how to ask well. Free, yours, run with it.

The engagement is behavioral strategy, three pillars:

1. **Authority.** How to become the obvious choice in your space instead of one of the options.
2. **Decision making.** The psychology behind how your buyers, your team, and your partners actually decide.
3. **The bottom line.** Learning what genuinely moves revenue, then scaling on behavior instead of guesswork.

THE PROMISE LINE

I train you and your team to be serious contenders in any industry, and I build you into a proper CEO.

THE QUALIFICATION, AND DO NOT SOFTEN IT

"This is for people running a team, with the business doing $200k or more. Below that line, everything you need is already in your hands. Today's material plus your own effort gets you to $200k, and I would rather you keep your money and do it yourself. I am worth hiring at the point where the bottleneck stops being effort and starts being strategy."

WHY SAYING THAT OUT LOUD WORKS

It disqualifies the wrong people in public, which is the single most credible thing you can do in a room like this. Nobody selling a course tells you not to buy yet.

THE CLOSING LINE

Your business should be outperforming everyone in your category. Let's make it.

Contact to put on screen: chris@bookchrismichael.com and chrismichael.site/#consult. Both are on the last page of their handout.

16 Prep Checklist and Likely Questions

RUN THIS BEFORE YOU START

BEFORE YOU OPEN YOUR MOUTH

- [ ] Terminal open with `claude` ready in a demo folder
- [ ] A throwaway or low-stakes site repo for the Netlify demo, not a revenue surface
- [ ] Netlify dashboard logged in, deploys tab open
- [ ] github.com/chrismikemagic/claude-toolkit open in a tab
- [ ] iMessage demo pointed at self-chat or a test contact only
- [ ] Notification-heavy apps closed, anything with client names off screen
- [ ] Install one-liner ready to paste: `curl -fsSL https://claude.ai/install.sh | bash`
- [ ] Type `/model` live to show the picker and `/effort` to show the dial (30 seconds, very concrete)
- [ ] claude.ai/code open in a tab, Claude app ready on your phone for the cloud session and push notification
- [ ] Optional: a Gmail draft demo, Claude reads an inquiry and you show the draft sitting unsent

LIKELY QUESTIONS, SHORT ANSWERS

Is it safe to let Claude change my files? Git. Every change is a save point and everything is reversible. It also asks permission before anything state-changing.

How much does it cost? Included in any paid Claude plan, Pro on up. Usage limits vary by plan.

Do I need to know how to code? No. You need to know what you want and how to check it got done. Claude writes the code.

What if Claude does something wrong in my account? Scoped tokens and allowlists. Give it the minimum access, and you can revoke a token instantly.

Windows? Yes, one-line PowerShell install. iMessage integration is Mac-only.

Which plan should I start with? Pro to learn. Upgrade the day hitting the cap costs you more than the upgrade does.

Does my computer have to stay on? No. Cloud sessions and routines run on Anthropic's machines; start and monitor them from your phone.

Can Claude really send email by itself? Yes, with your own token or Apple Mail. The official connector stops at drafts on purpose. Start draft-first and automate one category at a time.

Where are the docs? code.claude.com/docs. Setup, skills, and MCP all live there.

THE PART I DID NOT TEACH TODAY

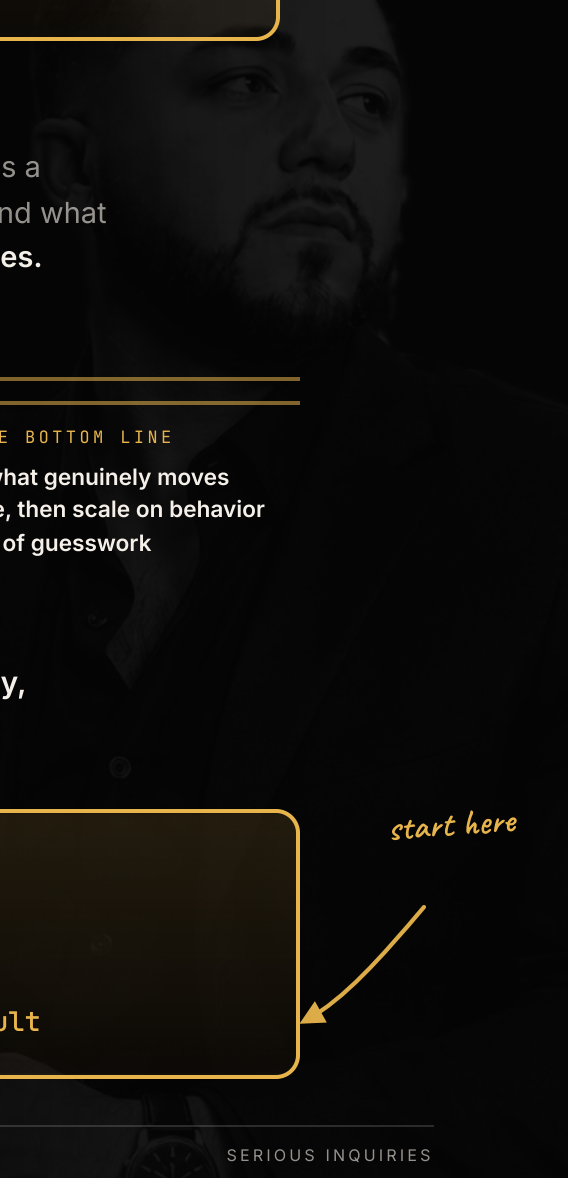
What I Build For Companies, I Can Build For You

read this part before anything else

WHO THIS IS FOR

If you run a team, and the business is doing $200k or more.

Below that line, what you already have is enough. **Today's material plus your own effort gets you to $200k, and I would rather you keep your money.** Hire me when the bottleneck stops being effort and starts being strategy.



Today was tools, and tools are the easy part. What decides who wins a category is behavior: how you build authority, how people decide, and what actually moves your bottom line. **That is the work I do with companies.**

01 AUTHORITY

Become the obvious choice in your space instead of one of the options

02 DECISION MAKING

The psychology behind how your buyers, your team, and your partners actually decide

03 THE BOTTOM LINE

Learn what genuinely moves revenue, then scale on behavior instead of guesswork

I train you and your team to be serious contenders in any industry, and I build you into a proper CEO.

start here

Your business should be outperforming everyone in your category. Let's make it.

EMAIL ME

chris@bookchrismichael.com

CONSULTING

chrismichael.site/#consult