NEVER HIT YOUR CLAUDE USAGE LIMIT AGAIN.

The 5-layer system I use to run a 30-person AI company on Claude — without going broke.

MILES DEUTSCHER

25 / Australia / Founder & CEO, AI Edge
Building with AI daily since ChatGPT launched
Running a 30-person AI media & software company

READ THIS FIRST

The limit isn't your problem.

Most people hit Claude's usage limit by lunchtime and assume they need to upgrade. They don't. They need a better system.

I use Claude 12 hours a day across my company, Claude Code sessions, agents running 24/7, and personal projects. I almost never hit limits. Not because I spend more — because I structured my workflow around the five levers that actually move the needle.

This document is the full framework. Five layers, stacked in order of impact. Layer 1 alone will double your effective usage. Layer 4 is how I scale past human bandwidth. Layer 5 is the part Anthropic doesn't advertise.

THE FIVE LAYERS

01	**Workflow + Mindset**	How you prompt
02	**Model Selection**	What you prompt with
03	**Tool Splitting**	Where you prompt
04	**Delegation**	Scaling past yourself
05	**Subscription Truth**	The honest math

LAYER 01

Workflow + Mindset

This layer is 80% of the game. Master it and you may never need the other five.

— Plan before you prompt
Claude talking is cheap. Claude building is expensive. Most people figure out what they want while prompting — that's where tokens die. Write the intent down in one line before opening a chat.

— Ask for clarifying questions
Before any large task: 'Ask me 5 clarifying questions before building anything.' One good build beats six rebuilds. This single prompt saves hours.

— Control output length
Most people let Claude default to 2,000 words when they need 100. 'One paragraph.' 'Five bullets.' 'Checklist format.' Massive lever nobody talks about.

— Long chats are a tax
Every message re-sends the full history. 50 turns deep = roughly 10x the cost of a fresh chat. Context carry-over feels productive. It isn't. Start new chats aggressively.

— Parallel beats long
Three focused chats on three problems beats one mega-chat juggling all three. Context bloat is exponential. Split, don't stack.

LAYER 01 / CONTINUED

— Projects over re-pasting
Stop re-pasting the same documents every chat. Load them into a Project once. Reuse forever. Project content is cached and doesn't count against your limit when reused.

— Attach files, don't paste them
Attached files sit in one place. Pasted text sits in every message going forward. Paste a 10-page doc into a 50-turn chat — that's the doc sent 50 times.

— Turn off extended thinking for simple tasks
Extended thinking burns tokens. Leave it off for routine work. Turn it on only when the task genuinely requires deep reasoning. Most people leave it permanently on and wonder why they hit walls.

— Build a memory file system
Keep a folder of instructions Claude auto-loads at the start of every chat. Self-updating — it writes back to the file as sessions end. Next session starts from where you left off. I never re-explain context. This one system saves me ~30% of my usage.

— In Claude Code: /compact and /clear
/clear wipes the chat. /compact summarizes it and keeps going. Use /clear when switching tasks, /compact when continuing a long one. The single most effective lever in Claude Code.

> **THE ROOT CAUSE**
> *Most 'I hit my limit by lunchtime' reports trace back to one of these: long wandering sessions, Opus left on by default, or huge pastes that sit in context forever. Fix those three and the limit feels very different.*

LAYER 02

Model Selection

Using Claude — or Opus — for everything is the single biggest cost leak.

INSIDE CLAUDE

— Haiku
Tweet rewrites, summaries, quick formatting, brainstorming, mechanical edits, renaming, boilerplate. Fast, cheap, capable enough for all of the above.

— Sonnet — the default
Most real work. Coding, writing, analysis, strategy. This is what you should be running by default unless the task genuinely demands more.

— Opus — the scalpel
Complex reasoning, large refactors, genuinely hard debugging, architectural calls. Uses meaningfully more quota. Switch to it when you need it. Don't leave it on.

OUTSIDE CLAUDE

— Don't just use Claude
Claude is not the right tool for everything. For bulk research, scraping, data cleaning, and first-draft thinking — other AI tools are faster, cheaper, or free. Use them for grunt work. Pass the cleaned output to Claude for the polished final.

— The stack I actually use
Perplexity for live research and citations. ChatGPT for quick lookups and image generation. Gemini for long-document analysis when I've already burned my Claude context. Open-source models (Llama, Mistral) for scraping and data cleaning at scale.

— The rule
Claude for the final 20% — synthesis, writing, decisions. Everything else runs on cheaper tools. I offload ~80% of upstream work. Claude only touches the output.

LAYER 03

Tool Splitting

Claude isn't one product. It's a stack. Most people use 20% of it.

— Claude Code with an API key = separate bucket
If you sign in to Claude Code with your plan, it eats from the same bucket as chat. Run Claude Code with an API key instead and it becomes pay-per-token with no plan cap. My setup: Max for chat, API key for heavy coding days. Effectively doubles my ceiling.

— Cowork for async work
Background tasks that don't need babysitting. Research runs, document cleanup, anything that can happen while you're working on something else. Free up your attention AND your context.

— Skills — the biggest unlock
Skills bundle your instructions into a file Claude auto-loads only when triggered. Write the rules once. Stop rewriting prompts every chat. Two wins at once: save context (they don't sit in every message) and save time (no re-explaining style, format, or rules).

— Artifacts — iterate in place
Stop regenerating documents from scratch. Iterate inside the Artifact. 30 revisions on one doc inside one Artifact costs roughly the same as a single generation.

LAYER 04

Delegation

One human hits limits. Five agents sharing the load don't.

— The math changes

Your output stops being constrained by how fast you can type or how many turns you get. It starts being constrained by how well you design your agents. Limits stop being a ceiling. They become irrelevant.

— My setup — OpenClaw

Five AI agents running 24/7 on dedicated Mac Minis. Each has its own context, its own task, its own budget. Oscar orchestrates. Samuel researches. Aria runs content strategy. Cyrus handles engineering. Roman runs quant trading analysis.

— You don't need five

Start with one. One Claude Code instance running a background task while you work in chat. Scale from there. The pattern matters more than the headcount.

> **THE OPERATOR MINDSET**
> *Stop thinking of Claude as a chatbot. Start thinking of it as a workforce. You're not a user. You're a manager. The limit isn't how much you can consume — it's how much you can orchestrate.*

LAYER 05

Subscription Truth

The part Anthropic doesn't put in the marketing.

— The plans

Pro — entry level. Max 5x — roughly 5x the usage of Pro. Max 20x — roughly 20x. Verify current ratios on the pricing page before you commit, because these move.

— The trend

Claude is getting less democratic, not more. Features that used to be accessible on lower tiers are migrating to paid plans. If you're serious about AI, the cost of entry keeps rising. Plan accordingly.

— My stack

Max 20x for chat and Claude Code — daily driver. API key on top for caching, batch, and agent workloads. Total: less than one junior employee salary. Output: more than five.

> **THE HONEST MATH**
>
> *If you mastered Layers 1–4, Max 20x buys you more than most people extract from $1,000+ of unoptimized usage. $200 a month for what replaces an entire role isn't a cost — it's the deal of the century. Just don't pay it before you've done the work to earn it.*

QUICK REFERENCE

The 60-second checklist.

01 Plan the prompt before opening the chat
02 Ask for clarifying questions before any big build
03 Specify output length. Always.
04 Start a new chat when the task changes
05 Run parallel chats, not one mega-chat
06 Default to Sonnet. Haiku for grunt. Opus rarely.
07 Use other AI tools for scraping, research, first drafts
08 Use Projects for repeated context
09 Attach files. Don't paste them.
10 /compact or /clear in Claude Code every ~20 turns
11 Build your memory file system
12 Delegate the boring stuff to cheaper models or agents