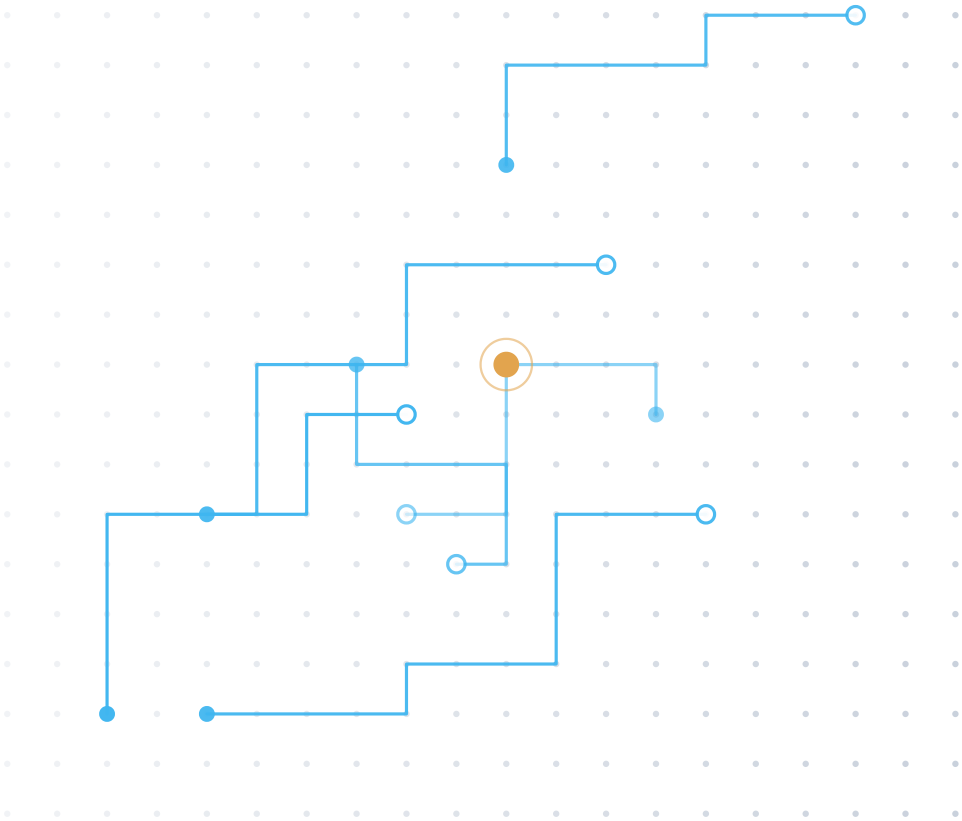


PRACTICAL GUIDE

The right model for each task: pay for judgment, not for reading

Split AI work across four tiers and spend less while getting more right

AUTHOR
Mario Chasco · Flowara AI

FOR
Technical teams and business owners who pay the AI bill

READING
9 min

EDITION
v2.0 · November 2026

BEFORE YOU START

What you will get from this guide

01 Decide which model does each task, based on how much reading it needs and what a mistake costs.

02 Set up an orchestrator and four agents with clear jobs and written limits.

03 Work out what one conversation costs before you price a monthly plan.

04 Write briefs that an agent understands the first time.

05 Check with real cases that your split is right, instead of assuming it.

CONTENTS

01 The problem: a senior architect at the photocopier
02 Four tiers, four jobs
03 The orchestrator reads, decides, delegates and verifies
04 Which task goes to which tier
05 What each tier costs
06 Five rules for splitting the work
07 Example: a new feature from start to finish
08 Checklist for splitting by tiers

For: Technical teams and business owners who pay the AI bill · 9 min read

01

The problem: a senior architect at the photocopier

Picture an architecture practice where the senior architect, the one with the highest hourly rate, spends the morning making photocopies, digging plans out of the archive and typing up meeting notes. The practice pays an expert's rate for an intern's work. On top of that, she arrives tired at the one decision that really needed her judgment.

AI models work the same way. Using the most powerful one for everything looks like the careful choice, but you pay for it twice. First on the bill: reading a hundred files to find three lines is the most expensive task and the one that needs the least judgment. Then on quality, because a context full of irrelevant reading leaves less attention for what matters, and mistakes hide in the noise.

TABLE 1 **The same work, two ways to pay for it**

	Everything on the top model	Split by tiers
BILL	High rate on every read	High rate only where there is risk
QUALITY	Context full of noise	Each model sees only its part
SPEED	Slow even on easy work	Mechanical work goes to the fastest
MISTAKES	Hidden among the reading	A verifier looks for them

The opposite does not work either. Asking a small model to design the architecture of a payments system saves a few cents and costs weeks. The way out is to split the work by tiers, as any well-run team does.

> *Pay for judgment, not for reading: every task goes to the lowest tier that does it well.*

THE RULE I USE TO SPLIT WORK AT FLOWARA AI

02

Four tiers, four jobs

The split has four roles. Each one maps to a model and a type of work, and each one has things it never does. This is how I have them set up on my server:

01 **Junior, on Haiku 5.5.** Discovery and summaries: finding files, reading logs, counting, sorting messages or summarising a long document. It only reads; it never edits and never gives an opinion on design.

02 **Builder, on Sonnet 5.5.** Routine implementation and writing once the design is decided: a new page from a template, one more test, documentation for something that already works. It does not settle design or security questions.

03 **Senior, on Opus 5.5.** Architecture, hard bugs, anything involving money or security, and any decision that is hard to undo. It is not used to search for files.

04 **Verifier, on Sonnet 5.5.** It reviews every significant delivery and answers *approved*, *approved with warnings* or *not approved*. It never fixes anything; if it did, it would stop being independent.

> ✦ IDEA
>
> The verifier is not a luxury. Whoever wrote something is the worst reviewer of their own work, person or model.

IN ONE LINE

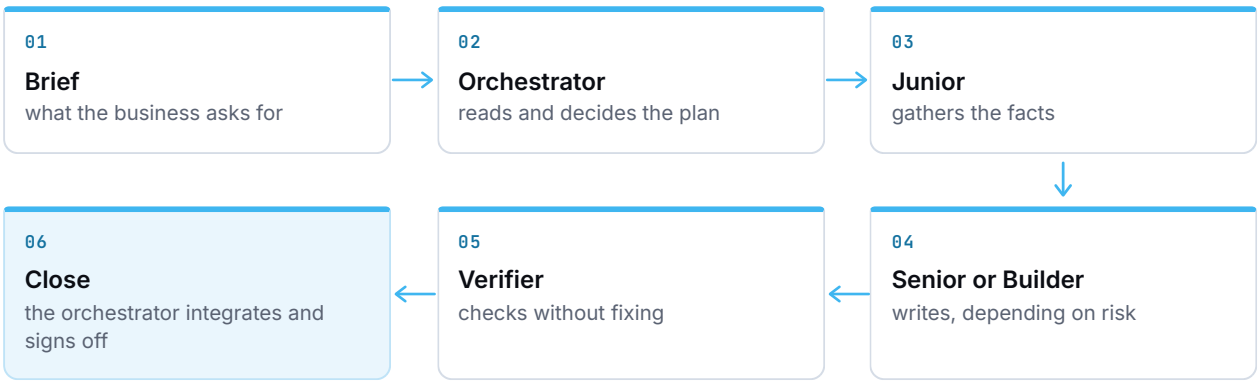
Each tier has one job and a written list of things it does not do.

03

The orchestrator reads, decides, delegates and verifies

Above the four tiers sits a main session that acts as the orchestrator. Think of a site manager on a building project: she understands the brief, decides the plan, hands out the pieces, reviews what comes back and puts the result together. What she does not do is write code or long documents herself, even when it looks like "just one line".

FIGURE 1 **How a brief moves through the tiers**



In my setup the main session runs on Fable 5.1, the most capable model in the family and also the most expensive. That is exactly why it does not write. Every line it drafted would cost five times what it costs on Sonnet 5.5, and its context is the most valuable thing in the system: fill it with code and it loses sight of the whole job.

There are two more reasons. Delegating forces you to write a clear brief, and that alone removes a lot of mistakes. And keeping the one who writes apart from the one who decides means every result passes through at least two pairs of eyes.

Independent tasks run in parallel. Tasks that touch the same files run one after another, with a checkpoint in between.

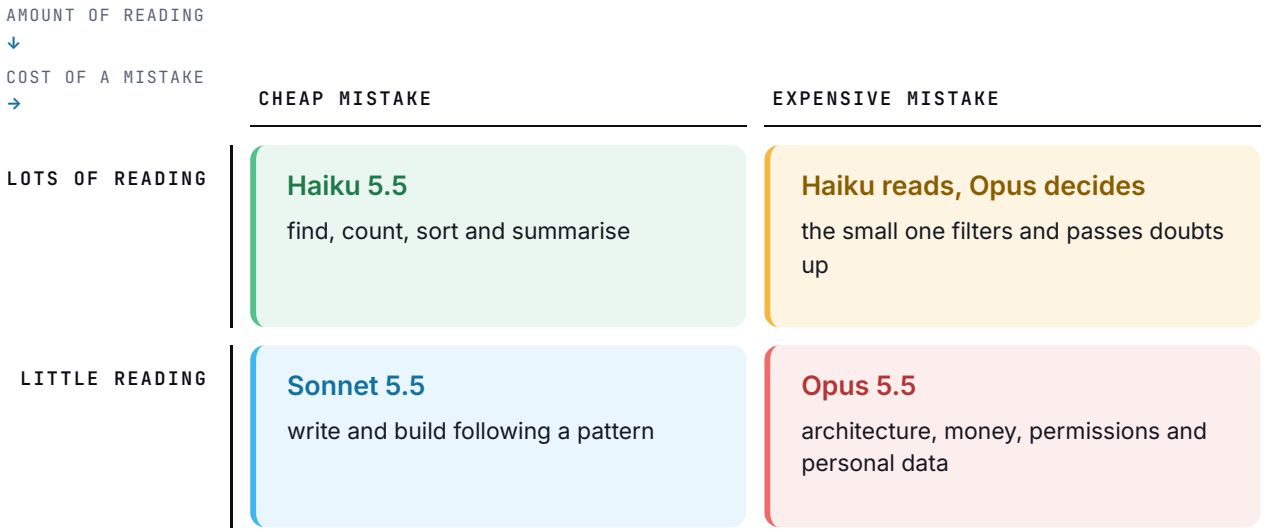
IN ONE LINE

The orchestrator does not write; it hands out the work and signs off the result.

04 Which task goes to which tier

When in doubt, ask yourself two questions. How much reading does this need? And how much does it cost to undo a mistake here? Lots of reading and little risk goes down. Little reading and lots of risk goes up.
For the second question I use a traffic light. If the mistake can be undone with one click, Green. If it costs an afternoon of work, Amber. If it touches money, permissions or personal data, Red, and then the task goes to Opus 5.5 however short it is.

FIGURE 2 **Which model for which task**



Notice that the top tier fills only one cell, but it is the cell that matters most. The amber cell pays off best: the cheap model does all the reading and the expensive one only sees what the cheap one is unsure about.

IN ONE LINE

Lots of reading goes down; lots of risk goes up.

05 What each tier costs

All four models accept a context of up to one million tokens, which is roughly 555,000 words. What changes is price and speed. These are Anthropic's public prices at the date of this guide:

MODEL	INPUT / OUTPUT, PER MILLION TOKENS	SPEED	WHAT I USE IT FOR
Haiku 5.5	from $0.10 / $0.50	Fastest	Sorting, counting, summarising
Sonnet 5.5	$2 / $10	Fast	Writing and building
Opus 5.5	$4 / $20	Moderate	Deciding and hard problems
Fable 5.1	$10 / $50	Slowest	Orchestrating, in my case

Two details bring the bill down a lot. Jobs that can wait, such as an overnight review, can go through batch processing at half the price. And anything repeated in every call, such as the rules or a business's catalogue, can be cached: reading it again costs 10% of the input price, and 5% on Opus 5.5 and Sonnet 5.5.

> ⚠ **WARNING**
>
> Treat these figures as an order of magnitude. Prices change, so check Anthropic's official pricing page before you quote anything: https://platform.claude.com/docs/en/about-claude/pricing

Before I price a monthly plan, I measure what one conversation costs. The sum is simple: input tokens times their price, plus output tokens times theirs. For example, a conversation with 2,000 input tokens and 300 output tokens (example figures, not from a client) costs less than one US cent on Sonnet 5.5, and twice that on Opus 5.5. It looks like nothing until you multiply it by a month of conversations at a hair salon with a busy WhatsApp.

To choose between two models for a specific task, I do not trust my instinct. I run the same real cases through both and note for each case whether it got it right and what it cost, with the seconds it took next to it. Sometimes the small model is just as accurate and answers sooner, and then there is nothing left to think about.

06 Five rules for splitting the work

These rules prevent almost every splitting mistake I have seen.

01 **Never move down a tier on judgment tasks.** If the decision is hard to undo or involves money or security, it goes to the top tier even if it is short. Saving there ends up costing more.

02 **Never move up a tier on mechanical tasks.** Searching, renaming, formatting or summarising does not get better on a pricier model; it only costs more.

03 **Digest the facts before you delegate.** The brief already carries what the agent needs: the files, the decisions already made, what is ruled out and what it must not touch. An agent that has to rediscover the context spends twice as much and gets less right.

04 **Ask for short reports.** At most 250 words covering what changed, how it was checked and what is still pending. A long report moves the cost of reading back to the orchestrator.

05 **Split into agents only what is large and divisible.** Each agent starts with its own context and reads again whatever it needs, so splitting multiplies token use. It pays off on large jobs that break into independent pieces; on a small change it costs more than doing it in one go.

> **EXAMPLE**
>
> Instead of "fix the form", the brief says: "the submission fails when the phone number is missing; this is the file; do not change the design; send me back the change and the test that proves it".

IN ONE LINE

A good brief is worth more than a pricier model.

07 Example: a new feature from start to finish

A travel agency wants to add a quote request form to its website that emails the team. This is how the work is split:

FIGURE 3 **A new feature, tier by tier**

1 **Orchestrator**
Reads the request and sets the scope: a form, an email alert and no payments.

2 **Junior on Haiku 5.5**
Finds the current contact form and how emails are sent today. Returns 150 words with the exact files.

3 **Senior on Opus 5.5**
Decides which personal data is stored, for how long and what is never written to the logs. A short task, but a judgment one.

4 **Builder on Sonnet 5.5**
Builds the form and the email template following that design, with its tests.

5 **Verifier**
Runs the tests and checks the logs. Answers approved with warnings: a friendly error message is missing.

6 **Close**
The builder adds the message, the verifier checks again and the orchestrator integrates.

The most expensive of the four tiers worked on a single piece, the one that carried risk. Everything else went to cheaper tiers, with an independent review at the end.

IN ONE LINE

Expensive judgment is spent where the risk is, not where the volume is.

08 Checklist for splitting by tiers

Use this list to review how your team, or your own setup, splits AI work.

Splitting by tiers, point by point

14 ITEMS

- [ ] Every task has a tier before it starts
- [ ] Anything hard to undo always goes to Opus 5.5
  money, permissions and security
- [ ] Bulk reading goes to Haiku 5.5
  finding, counting and summarising
- [ ] The orchestrator only reads, decides, delegates and verifies
- [ ] Every brief has a goal, files and what is ruled out
  and what not to touch
- [ ] Every brief sets the report format
  250 words at most

- [ ] A verifier who did not write it reviews every delivery
- [ ] The verifier reports and never fixes
- [ ] Same files, one after another
  with a checkpoint
- [ ] Parallel agents only for large, divisible jobs
- [ ] Cost per conversation measured before setting a plan price
- [ ] Models compared on real cases
  accuracy, cost and seconds
- [ ] Every agent has a named person who answers for it
- [ ] Every splitting mistake becomes a written rule

NEXT STEP

Free diagnostic (20 min)

If you want to see which part of your business can be automated with the risk under control, book a free 20-minute diagnostic.

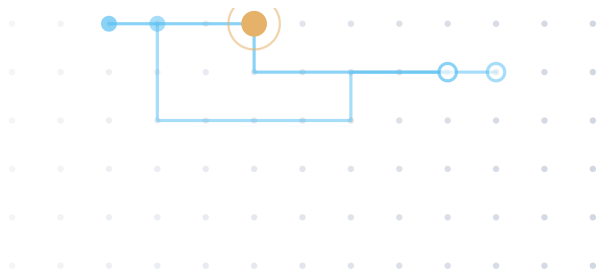
Book a diagnostic ›

flowaraai.com/en/contact

RELATED RESOURCES

Toolkit with the four agents for Claude Code ↗

Your Business Brain in Plain Text Notes ↗



Mario Chasco · Flowara AI

AI systems in production for small businesses.

flowaraai.com
github.com/mariochasco78557
youtube.com/@mariochasco_AI
linkedin.com/in/mario-adrian-chasco-frauca

Share this guide as is, crediting flowaraai.com.