PROMPTS & SKILLS FOR CLAUDE CODE

Sorting your work into job types and matching each to the tool that suits it.

Terminal Craft

INDEPENDENT FIELD GUIDE

Which AI For Which Job

Sorting your work into job types and matching each to the tool that suits it.

TERMINAL CRAFT

WELCOME

Before You Begin

Most AI stacks are archaeological rather than designed. One tool was good in March, another was recommended in June, a third arrived bundled with something else, and now there are five subscriptions and one of them does ninety per cent of the work regardless of whether it suits the task. Then, because that one tool is mediocre at half of what you push through it, you conclude that AI is disappointing. What happened is that you used a screwdriver on a nail for six months. This guide is a method for fixing that with evidence from your own work rather than from comparison articles that were written for someone else's job and will be out of date by spring. Inside: the four job types that cover nearly everything you do and why they want different things, a one-week sorting exercise that shows you your real distribution, a bench test that measures candidate tools on tasks where you already know the right answer, matching tables, an honest chapter on when a cheaper or smaller model is the correct call, and an audit that makes the cancellations obvious rather than agonising. Two notes. This is an independent publication with no affiliation to any AI vendor, and no software or subscription is included. And it is the 2026 Edition, which matters more here than in any

other guide in this system: specific model rankings move constantly, so this teaches you to run the test rather than naming winners that would be wrong within months.

— The Terminal Craft Team

HOW TO USE THIS GUIDE

Getting the Most From This Book

This is a reference, not a reading list. The prompts work best adopted a few at a time until they are automatic. Here's how to get the most from the pages ahead.

1. **Use it as a reference.**
   Every chapter ends with a *Try This* exercise and a set of *Key Takeaways*. Read the chapter once, then come back to the prompts when you need them.

2. **Adopt a few at a time.**
   Take four prompts and use them until they are automatic. Reaching for forty at once is the fastest way to use none of them.

3. **Read the 'not for' line.**
   A structured prompt on a trivial task is pure overhead. Knowing when not to use one matters as much as knowing the wording.

4 **Adapt, don't discard.**

When a model version changes and a prompt stops landing, rebuild it from the four-part frame rather than hunting for a new download.

Whenever you're ready, turn the page and let's begin.

CONTENTS

What’s Inside

CHAPTER 1

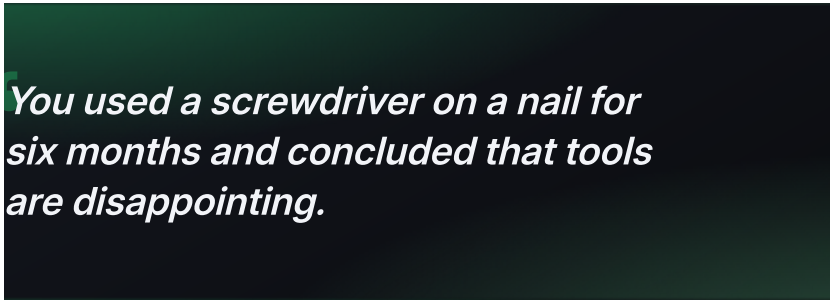
The Four Job Types

> Almost everything you ask an AI to do is one of four things, and they have different failure modes, different tolerances for error and different ideal tools. Sorting by job is what makes the rest of this possible.

Generate and transform

Generation is producing something new from a description: a function, a draft, a schema. The failure mode is plausible-but-wrong, and the cost of an error is usually rework rather than damage, because you review it before it matters.

Transformation is converting something that already exists: refactoring, translating, reformatting, migrating. The failure mode is silent loss: something dropped or subtly altered in the middle of a large change. That is much harder to catch by reading.



Verify and explore

Verification is checking work against a standard: reviewing a diff, auditing for a class of bug, confirming a claim. Here the failure mode is false reassurance, and it is the most expensive of the four because it removes a control you believed you had.

Exploration is building understanding: mapping a codebase, comparing approaches, working out what a system does. The failure mode is confident invention, and it is dangerous because you are asking about something you do not yet know.

Why the split matters

These want different things. Generation rewards fluency. Transformation rewards care and completeness over long inputs. Verification rewards scepticism and is actively harmed by agreeableness. Exploration rewards accurate retrieval and admitting ignorance.

No tool is best at all four, and the one you default to is almost certainly strong at one or two. The question is not which tool is best; it is which job you are currently doing.

SEEING YOUR REAL DISTRIBUTION

The Job Sort

For each of the last ten things I asked you, tag it: generate, transform, verify, or explore. Give me the counts and tell me which type dominates.

WHY IT WORKS

People are consistently wrong about their own distribution, and the counts usually reveal that verification is under-used.

USE WHEN Before changing any tool **PAIRS WITH** The Bench Test

A first session with no history to sort.

CHOOSING BEFORE YOU ASK

The Type Check

Before I ask this: which of the four job types is it, generate, transform, verify or explore? Name it and what would count as failure for that type.

WHY IT WORKS

Naming the failure mode in advance changes how carefully you read the answer, which is most of the benefit.

USE WHEN Any consequential request **PAIRS WITH** The Job Sort

Routine work where the type is obvious.

Try This: Tag One Week

The distribution surprises almost everyone.

1. For seven days, tag every AI task with one of the four types.
2. Count them at the end of the week.
3. Note which type you barely use. For most people it is verify.
4. Do not change any tool yet. This week is measurement.

KEY TAKEAWAYS

- Generate, transform, verify, explore. Four types, four different failure modes.
- Verification is the most expensive to get wrong, because it removes a control you trusted.
- No tool is best at all four, and the one you default to is strong at one or two.

CHAPTER 2

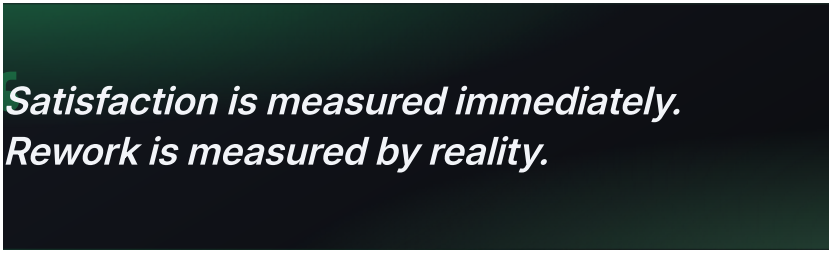
Sorting Your Own Week

> One week of tagging tells you more than any comparison article, because it is about your work rather than a benchmark suite.

Tag, do not change

The rule for the sorting week is that you change nothing. Same tools, same habits, same defaults. You are measuring a baseline, and adjusting while measuring produces a number that describes neither the old way nor the new one.

This is the discipline people skip, and skipping it is why most tool decisions are made on impressions rather than evidence.



Record three things per task

The type, the tool, and whether you had to redo it. That is enough. Adding more columns feels rigorous and produces a

log nobody fills in past Wednesday, which is worse than a rough log completed all week.

The redo column is the important one. It is a far better signal than satisfaction, because satisfaction is measured immediately and rework is measured by reality.

Read the distribution honestly

Two patterns are common. Most people do much more transformation than they think and far less verification than they should. And most people push every type through one tool regardless of fit.

Both are fixable, and neither is visible without the week of tagging. That is the entire argument for doing it before touching your subscriptions.

SETTING UP THE MEASUREMENT

The Week Log

Set up a three-column log for me: job type, tool used, and whether I had to redo it. Give me the format and one worked example row.

WHY IT WORKS

A log with three columns gets completed; one with seven does not, and an incomplete log is worse than a rough one.

USE WHEN Day one of the sorting week PAIRS WITH The Job Sort

Weeks that are wildly unrepresentative of your normal work.

DAY EIGHT

The Distribution Read

Here is my week of tagged tasks. Give me the counts per type, the redo rate per type, and the single clearest mismatch between what I am doing and what I am using.

WHY IT WORKS

The redo rate per type is where a mismatch becomes visible, and it is invisible in aggregate satisfaction.

USE WHEN After a full week of tagging

PAIRS WITH The Week Log

Partial weeks, where the counts mislead.

THE JOB YOU ARE NOT DOING

The Missing Type

Which of the four types barely appears in my log? What would it look like if I did that type deliberately, and what would it have caught this week?

WHY IT WORKS

The absent type is usually verification, and naming what it would have caught makes the gap concrete rather than theoretical.

USE WHEN After the distribution read

PAIRS WITH The Distribution Read

Logs that already show a balanced distribution.

Try This: Three Columns Only

The log you finish beats the log you designed beautifully.

1. Make a note with three columns: type, tool, redone?
2. Fill one row per task for seven days, taking under ten seconds each.
3. Change absolutely nothing else during the week.
4. On day eight, run the Distribution Read before any decisions.

KEY TAKEAWAYS

- Change nothing during the measurement week or the baseline is worthless.
- Three columns: type, tool, redone. The redo column carries the signal.
- Most people transform more than they think and verify far less than they should.

CHAPTER 3

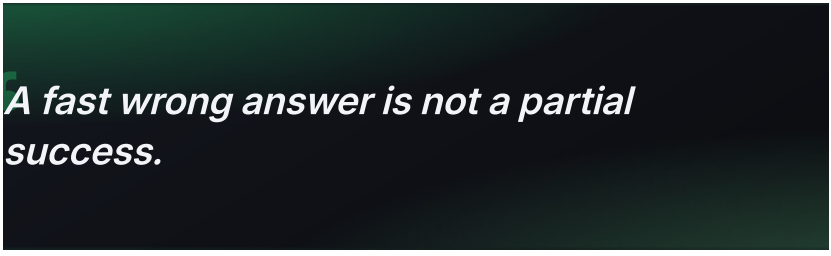
The Bench Test

> Published benchmarks measure someone else's tasks. This is a one-afternoon procedure that measures yours, and it is the only comparison that should change what you pay for.

Use tasks where you know the answer

Pick three real tasks from your own history where you already know what a correct result looks like. Ideally ones you have completed, so the answer is not in dispute. Novel tasks cannot be scored, and unscored comparisons collapse into which output read most pleasantly.

Three is enough to distinguish tools, and few enough that you will actually run it on more than one candidate.



Score correctness and rework only

Two numbers per run: was the result correct, and how much did you have to redo. Do not score tone, speed of typing, or how confident it sounded. Those are the things that make a tool feel good and have almost no relationship to whether it saved you time.

Speed is worth recording separately if it matters to you, but never mixed into the same score. A fast wrong answer is not a partial success.

Run the same task on each candidate

Identical wording, identical context, fresh session for each. Any difference in the prompt invalidates the comparison, and it is remarkably easy to unconsciously write a better prompt for the tool you already prefer.

Write the three prompts out first, before you touch any tool, and paste them unchanged.

CHOOSING THE THREE TASKS

The Bench Setup

From my recent work, pick three completed tasks where the correct result is unambiguous and I can score an answer objectively. One per job type I actually use most.

WHY IT WORKS

Scorable tasks are the whole basis of the test, and picking them from completed work removes any dispute about the right answer.

USE WHEN Before evaluating any tool **PAIRS WITH** The Score Sheet

Work where correctness is subjective.

KEEPING THE COMPARISON HONEST

The Score Sheet

Build me a score sheet: task, tool, correct yes/no, minutes of rework. No column for tone, confidence or writing quality.

WHY IT WORKS

Omitting the subjective columns is the point. They are what make a tool feel good while saving no time.

USE WHEN Before the first run **PAIRS WITH** The Bench Setup

Evaluations where writing quality is the actual product.

REMOVING YOUR OWN PREFERENCE

The Blind Compare

Here are three outputs for the same task, labelled A, B and C, with no tool names. Score each for correctness and name what you would have to fix.

WHY IT WORKS

Labels move judgement more than anyone expects, and stripping them is free.

USE WHEN After collecting the runs **PAIRS WITH** The Score Sheet

Comparisons where the outputs are trivially distinguishable.

Try This: One Afternoon

Three tasks, two tools, one honest answer.

1. Pick three completed tasks with unambiguous correct results.
2. Write the three prompts out and do not adjust them per tool.
3. Run all three on each candidate in fresh sessions.
4. Score correctness and rework only, then read the totals.

KEY TAKEAWAYS

- Use tasks you have already completed, where the right answer is not in dispute.
- Score correctness and rework. Never tone, confidence or perceived speed.
- Write the prompts before touching a tool, or you will write a better one for your favourite.

CHAPTER 4

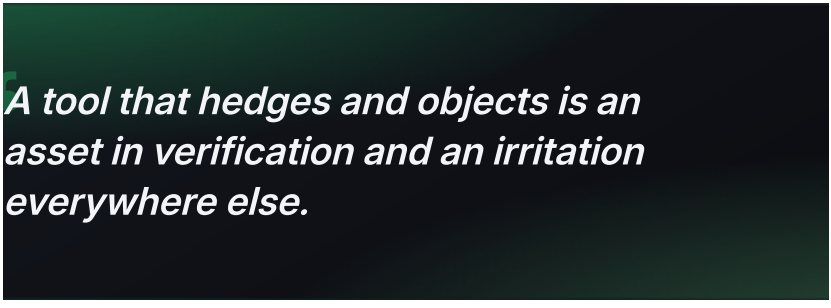
Matching Jobs To Tools

> With a distribution and a bench result, the matching is mostly mechanical. What follows is how to reason about it, not a list of product names that would be stale before you read them.

What each type actually needs

Generation wants fluency and breadth, and tolerates a tool that occasionally overreaches, because you are reviewing it anyway. Transformation wants care over long inputs and completeness: the ability to carry a large file through a change without dropping something in the middle.

Verification wants the opposite of agreeableness. A tool that hedges and objects is an asset here and an irritation everywhere else, which is exactly why one default tool serves you badly.



Exploration wants honesty about gaps

For understanding an unfamiliar system, the most valuable property is a willingness to say 'I cannot find that' rather than producing a plausible answer. Retrieval accuracy matters more than reasoning brilliance, because you cannot check what you do not yet know.

This is the type where the bench test is most worth running with citation requirements, since a tool's citation accuracy is measurable in a way its 'intelligence' is not.

Two tools is usually the right number

Most people end up with a primary for generate and transform, and a second one for verify. Different specifically so its blind spots differ. Exploration usually goes to whichever has better access to your codebase.

Three or more is worth having only when the bench test shows a clear, repeated win for a third on a type you do a lot of. Otherwise it is subscription drift with extra steps.

TURNING RESULTS INTO ASSIGNMENTS

The Matching Table

Here is my type distribution and my bench scores. Assign a tool to each job type and justify each assignment in one line from the scores, not from reputation.

WHY IT WORKS

Forcing the justification to come from your own scores keeps reputation and marketing out of the decision.

USE WHEN After the bench test **PAIRS WITH** The Overlap Audit

Before you have any scores to reason from.

CHOOSING YOUR VERIFIER DELIBERATELY

The Contrarian Pick

For verification I want the tool least similar to my primary. Which of my candidates disagreed with the primary most often in the bench runs, and on what kind of point?

WHY IT WORKS

A verifier's value comes from having different blind spots, so similarity to the primary is a disqualification rather than a comfort.

USE WHEN Choosing the second tool **PAIRS WITH** The Matching Table

Setups with only one tool available.

CHECKING WHETHER A THIRD IS JUSTIFIED

The Two-Tool Test

Would dropping my third tool cost me anything measurable, based on the bench scores and my type distribution? Argue for dropping it.

WHY IT WORKS

Arguing for the drop surfaces the honest answer faster than asking whether to keep it.

USE WHEN Any stack of three or more **PAIRS WITH** The Matching Table

Stacks of two that are already lean.

Try This: Assign On Paper

Write the assignment down or it will not survive contact with habit.

1. Write the four job types in a column.
2. Next to each, write the tool your bench scores support.
3. Where the answer is the same tool for all four, ask whether you tested a different candidate.
4. Put the list where you work, and follow it for two weeks.

KEY TAKEAWAYS

- Generation tolerates overreach; verification is actively harmed by agreeableness.
- For exploration, a tool that admits gaps beats one that reasons impressively.
- Two tools is usually right: a primary, and a deliberately different verifier.

CHAPTER 5

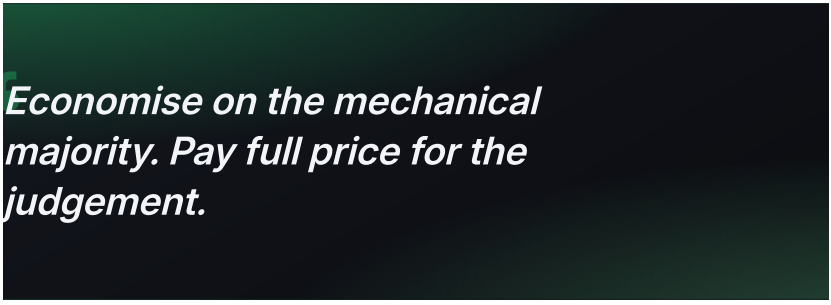
When Smaller Is Better

> There is a strong pull toward always using the most capable option. For a meaningful share of your work it is the wrong call, and noticing which share is worth real money.

Mechanical work does not need reasoning

Reformatting, renaming, mechanical translation, extracting a list, applying a known pattern across many files. These are transformation tasks with a known answer, and a smaller, faster model does them at the same correctness for a fraction of the cost and latency.

The test is whether the task requires judgement. If you could specify it completely, you are paying for reasoning you are not using.



Latency is a real cost

For anything in a tight loop, a fast answer that is right is worth more than a slower one that is also right. Quick questions, repeated small transforms, anything where you are waiting. Time spent waiting is time spent losing the thread.

This is the argument people most often miss, because cost is visible on an invoice and latency is only visible in how the day felt.

Where to never economise

Verification, anything security-related, and anything where being confidently wrong is expensive. Those are the jobs where the difference between models shows up most, and where the saving is trivial next to the cost of the failure it enables.

Economise on the mechanical majority. Pay full price for the judgement.

DECIDING IF A TASK NEEDS JUDGEMENT

The Specification Test

Could this task be specified completely enough that the answer is determined? If yes, it does not need reasoning. Say so and give me the complete specification.

WHY IT WORKS

A fully specifiable task is one you are overpaying for, and producing the specification proves the point rather than asserting it.

USE WHEN Repetitive or mechanical work

PAIRS WITH The Downgrade Trial

Tasks requiring genuine design decisions.

TESTING A CHEAPER OPTION SAFELY

The Downgrade Trial

I will run these five mechanical tasks on a smaller model. Tell me exactly what to check in each result to catch a quality drop I would otherwise miss.

WHY IT WORKS

Knowing what to check in advance is what makes a downgrade a test rather than a gamble.

USE WHEN Before moving work to a cheaper tier

PAIRS WITH The Specification Test

Verification or security work.

PROTECTING THE JOBS THAT MATTER

The Never List

Given my work, list the categories where I should never use a cheaper or smaller option, and give a one-line reason for each.

WHY IT WORKS

A written exclusion list stops cost optimisation from quietly reaching the jobs where being wrong is expensive.

USE WHEN Once, when setting up tiers

PAIRS WITH The Downgrade Trial

Setups with only one tier available.

Try This: Move Five Tasks Down

A small, reversible experiment with a real answer.

1. Pick five fully specifiable tasks from your log.
2. Run them on the cheapest option you have access to.
3. Check the specific things the Downgrade Trial named.
4. If quality held, move that whole category permanently.

KEY TAKEAWAYS

- If you can specify it completely, you are paying for reasoning you are not using.
- Latency is a real cost and only shows up in how the day felt, never on the invoice.
- Never economise on verification, security, or anything where confident errors are expensive.

CHAPTER 6

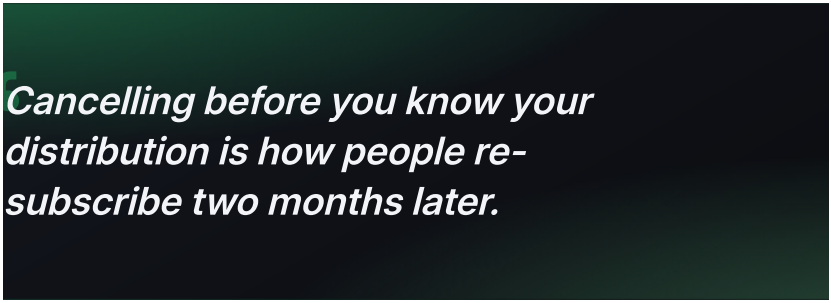
The Overlap Audit

> The last step is the one that saves money, and it is deliberately last: cancelling before you know your distribution is how people end up re-subscribing two months later.

One unique job each

For every tool you pay for, name the one job it is uniquely best at according to your own bench results. Anything without a unique job is a candidate for cancellation, and the exercise is uncomfortable because it is clear.

The common outcome is that two tools claim the same job and one of them is kept out of habit or sunk cost.



Cancel one, not all

Drop a single subscription and work for a month. If nothing degrades, drop the next. Cancelling three at once means you cannot attribute a problem to any of them and will probably re-subscribe to all three in a panic.

It is the same one-change-at-a-time discipline as everything else in this system, applied to spending.

Re-run this quarterly

The landscape moves fast enough that a decision made in March is a hypothesis by September. Re-run the bench test quarterly, and only change something if the result is clear. Not the whole exercise, just the three tasks.

A quarterly fifteen-minute test is what keeps a stack deliberate. Reading release announcements is what keeps it archaeological.

FINDING WHAT YOU ARE PAYING TWICE FOR

The Overlap Audit

List every AI tool I pay for and the one job each is uniquely best at from my bench results. Flag any tool without a unique job.

WHY IT WORKS

Uniqueness is a much clearer criterion than usefulness, and almost everything is useful.

USE WHEN Quarterly **PAIRS WITH** The Cancel Trial

Stacks of one or two.

DROPPING ONE SAFELY

The Cancel Trial

I am dropping this tool for a month. What specifically should I watch for that would tell me it was actually load-bearing?

WHY IT WORKS

Naming the signal in advance stops the decision being made on vague discomfort a week in.

USE WHEN After the audit flags a tool **PAIRS WITH** The Overlap Audit

Tools with contractual or team-wide commitments.

KEEPING THE DECISION CURRENT

The Quarterly Re-Bench

Re-run my three bench tasks against my current tools and any new candidate. Tell me only whether the ranking changed, and by how much.

WHY IT WORKS

Restricting the output to whether the ranking changed prevents interesting-but-irrelevant differences from triggering churn.

USE WHEN Every quarter **PAIRS WITH** The Overlap Audit

Months when you have no capacity to act on the answer.

Try This: Name the Unique Job

The uncomfortable question, asked once a quarter.

1. List everything you pay for.
2. Next to each, write the one job it is uniquely best at for your work.
3. Any blank is a cancellation candidate.
4. Cancel exactly one, and work for a month before touching the next.

KEY TAKEAWAYS

- Ask what each tool is uniquely best at. Usefulness is too generous a criterion.
- Cancel one and wait a month, or you cannot attribute what degraded.
- Re-bench quarterly. Reading release announcements is not a method.

PUT IT INTO PRACTICE

Your Action Plan

Two weeks from an archaeological stack to a deliberate one. Measure first, decide second, cancel last.

- [ ] Set up the three-column log: type, tool, redone.
- [ ] Tag every AI task for seven days and change absolutely nothing else.
- [ ] On day eight, read the counts and the redo rate per type.
- [ ] Identify the type you barely use. For most people it is verification.
- [ ] Pick three completed tasks with unambiguous correct results.
- [ ] Write the three bench prompts before touching any tool.
- [ ] Run all three on each candidate in fresh sessions, wording unchanged.
- [ ] Score correctness and rework only. No tone, no confidence, no perceived speed.
- [ ] Assign a tool per job type, justified from your scores rather than reputation.
- [ ] Choose your verifier for being different from your primary, not for being best.

- [ ] Move five fully specifiable tasks to a cheaper option and check what matters.
- [ ] Run the overlap audit, cancel exactly one, and re-bench next quarter.

FINAL WORDS

Where You Go From Here

The goal is not a bigger stack or a smaller one. It is a deliberate one, where you can say, for each tool you pay for, what job it is uniquely best at on your own work. That sentence is the whole test, and most people cannot say it about at least one of their subscriptions today. Do the measurement week before anything else, because every decision after it is better and none of them are possible without it. Keep the bench test to three tasks so you will actually re-run it. Choose your verifier for being different rather than for being best, since its value comes entirely from having other blind spots. Economise on the mechanical majority and never on the judgement. And accept that this is a quarterly exercise rather than a permanent answer. The landscape moves fast enough that a decision made today is a hypothesis in six months. Fifteen minutes every quarter keeps the stack deliberate. Nothing else does.

Breathe. Begin. Return.
