FREE STARTER GUIDE · PREVIEW OF THE AGENT WORK PROTOCOL

Claude Code Routing Quickstart

Route every coding task to the right model, the right files and the right review loop — so Claude Code ships changes you can trust.

For	Level	Time	Next level
Developers and technical builders	Intermediate	45 minutes for setup	Chapter 1 of the full playbook

THE MICRO-PAIN THIS GUIDE FIXES

Claude Code is fast. Unrouted, it is fast in the wrong direction.

You open Claude Code, describe a feature, and twenty minutes later you have a diff touching fourteen files, two new dependencies and a test suite that no longer runs. Nothing is exactly wrong — but nothing is clearly right either, and you now spend an hour reviewing work you could have scoped in five minutes.

14 files
A typical unscoped diff when the agent is given the whole repo and a vague goal.

1 hour+
Review time lost on changes that were never bounded by an acceptance test.

2×
Token spend when a heavy model is used for tasks a lighter setting handles fine.

When you finish this free guide you will have

- A routing card that decides model, context and review depth for any task in under a minute
- An acceptance check written before the agent opens a single file
- A reusable `CLAUDE.md` section that keeps the agent inside your boundaries
- A review loop that ends with a diff, a passing test and a written note of what is still uncertain

THIS FREE STARTER

- One working route for one problem
- 7 short steps and a copy-and-use prompt
- One worked example and a final checklist

THE FULL PLAYBOOK · $29

- 6 chapters that take the whole job end to end
- 10 ready-to-use templates, a worked case, 137 PDF pages
- **See The Agent Work Protocol →**

STEP BY STEP

The working method

Routing means answering four questions before the agent acts: how hard is the task, what context does it need, how will we know it works, and how carefully must a human review it.

You need: Claude Code installed and signed in (terminal, IDE extension or desktop); A Git repository with a working test command; Your editor for reviewing diffs. **Rule:** Always start on a clean branch. Commit before you ask for changes.

STEP 01 OF 07

Classify the task by shape

- Label the task: **mechanical** (rename, format, small fix), **feature slice** (new behaviour in known code), **investigation** (bug with unknown cause) or **architecture** (changes structure).
- Estimate blast radius: how many files and which layers could change?
- Write the shape and blast radius as the first line of your prompt.

Example: "Feature slice, low blast radius: add CSV export to the existing report page (2–3 files)."

Done when: The task has one label and an expected file count.

STEP 02 OF 07

Write the acceptance check first

- Write one observable behaviour that proves the task is done.
- Turn it into a command where possible: a test to add, a curl call, a CLI output.
- Add one negative case: what must **not** change.

Example: "Done when `pytest tests/test_export.py` passes and the existing report tests still pass."

Done when: You can run the check yourself in under a minute.

STEP 03 OF 07

Pick the model and effort setting

- Mechanical tasks → default or faster model settings.
- Feature slices → default model; ask for a short plan first.
- Investigations and architecture → the most capable model available to you, with an explicit planning step (e.g. plan mode) before any edit.
- Check your plan's current model options in Claude Code — they change over time.

Example: Investigation of a flaky test → strongest model, plan mode, no edits until the cause is written down.

Done when: The model choice is written next to the task label.

STEP 04 OF 07

Scope the context deliberately

- Name the 2–5 files that matter; mention them with @ or paths in the prompt.
- Add boundaries to `CLAUDE.md`: folders not to edit, commands to run, style rules.
- For investigations, allow reading widely but editing narrowly: "read anything, edit only src/export/".

Example: "Relevant: @src/reports/page.tsx, @src/reports/api.ts. Do not edit /migrations or package.json."

Done when: Your prompt lists the files in scope and the files out of scope.

> **In the full playbook ·** The Agent Work Protocol includes a full routing matrix for coding, research, data and writing tasks — with the review depth each one needs. See what is inside · Get it for $29

STEP 05 OF 07

Ask for a plan, then approve a slice

- Ask: restate the goal, list unknowns, propose the smallest slice and how it will be tested.
- Correct the plan in one message. Approve only the first slice.
- If the plan touches more files than your estimate, stop and re-scope.

Example: Plan proposes a new export library → you reply: "No new dependencies; use the existing CSV util."

Done when: The approved slice fits your blast-radius estimate.

STEP 06 OF 07

Run the check and read the diff

- Have the agent run the acceptance command and show the output.
- Read the diff yourself: `git diff` or your editor's review view.
- Ask the agent to list anything it assumed or could not verify.

Example: Test passes; diff shows 3 files; agent notes it did not test files over 10k rows.

Done when: You saw the test output and the diff, and the open questions are written down.

STEP 07 OF 07

Commit with a routing note

- Commit with a message describing behaviour, not files.
- Add a one-line routing note to your team doc: task shape → model → context → check → result.
- If the route failed, write why. Failed routes are the most valuable notes.

Example: "Feature slice / default model / 3 files / pytest export → worked first try."

Done when: Commit made, routing note saved, branch ready for review.

COPY AND USE

Prompts and template

Paste into your AI tool and replace everything in [brackets].

PROMPT 1 · ROUTED TASK BRIEF

```
Task shape: [mechanical | feature slice | investigation | architecture]
Blast radius: [expected files / layers]
Goal: [one sentence, behaviour-level]
Done when: [acceptance command] passes and [negative case] is unchanged.
Relevant files: [@path1, @path2]
Do not edit: [folders/files]

Before editing anything:
1. Restate the goal and the acceptance check.
2. List unknowns that could change the approach.
3. Propose the smallest first slice and how you will test it.
Wait for my approval before making changes.
```

WORKED EXAMPLE

Adding CSV export without a surprise refactor

Marco needs CSV export on a reporting page. Last time he asked Claude Code for a feature, it refactored the data layer.

1. Label: feature slice, 2–3 files. Acceptance: new test `test_export_csv` passes; existing report tests unchanged.
2. Model: default setting, plan first.
3. Context: @src/reports/page.tsx, @src/reports/api.ts, @src/utils/csv.ts. Do not edit /db.
4. Plan comes back proposing a new CSV library; Marco redirects it to the existing util.

RESULT

Feature shipped in 25 minutes with a 3-file diff, a passing test and a documented edge case — and a route the team reuses next time.

FINAL CHECKLIST

Before you call it done

- [ ] Clean branch and a commit before starting
- [ ] Task labelled by shape with a blast-radius estimate
- [ ] Acceptance check written and runnable
- [ ] Model/effort chosen for the task shape
- [ ] Files in scope and out of scope named
- [ ] Plan reviewed; only the first slice approved
- [ ] Test output seen, not just reported

In the full playbook · The full protocol adds planning cards that force the agent to list unknowns before it edits — the single habit that removes most rework. See what is inside · Get it for $29

WHEN IT GOES WRONG

SYMPTOM	LIKELY CAUSE	FIX
Diff is much bigger than expected	Scope not stated; agent 'improved' nearby code	Add file boundaries and the >N-files stop rule to CLAUDE.md
Agent says tests pass but CI fails	Wrong or partial test command	Put the exact CI test command in CLAUDE.md and require output
Session goes in circles	Investigation treated as a feature	Switch to investigation prompt: hypotheses before edits
Hitting usage limits early	Heaviest model on mechanical tasks	Route mechanical work to lighter settings

LOCKED IN THE FULL PLAYBOOK · TOOLKIT PREVIEW

This free starter shows the method. The Agent Work Protocol ships the ready-to-use files:

- AGENTS.md rules template
- Agent work protocol
- Autonomy levels matrix
- Compact system instruction
- Handoff memo template
- Implementation notes template

Unlock all templates — $29 one-time →

WHERE THIS FREE GUIDE STOPS

You fixed one problem. The full system fixes the job around it.

This starter gave you one route through one step. It deliberately leaves out what makes the result hold up at work:

- **The steps before and after this one.** The Agent Work Protocol covers the whole job in six chapters, not one slice of it.
- **Ready-to-use templates.** 10 copy-and-use files, so you do not rebuild the structure every time.
- **A full worked case** that shows every decision on a realistic project, start to finish.
- **Checks for when it fails**, so a result you ship tomorrow is still right next month.

YOUR NEXT STEP · THE AGENT WORK PROTOCOL

Make every agent task this predictable

What you will be able to do:

- Turn any vague request into a task brief an agent can execute without guessing
- Set the right level of agent autonomy by impact and reversibility
- Catch errors before handoff with a three-layer verification pass
- Hand work off with a memo the next person or agent can act on

6	10	1	137
chapters	templates	worked case	PDF pages

You just read a preview of **Chapter 1: Turn every request into a task brief before the agent touches anything**.

~~$49~~ **$29** one-time · instant access in your CosynkAI Library

Get the playbook — $29 one-time → ***See what is inside***

WORKING ON SEVERAL AI WORKFLOWS?

Complete Guide Library — all 8 CosynkAI playbooks for **$97** one-time ($472 combined regular prices). Already own a playbook? Its price is credited toward the Library.

Get the Library — $97 → Compare all 8 playbooks

You received this free guide after sending the keyword **CODING**. Developed by CosynkAI.com · Edition 1.0 · September 2026. Educational resource, not an implementation service. Tools, prices and limits change; check current documentation before relying on a tool in production. Questions: hello@cosynkai.com · Privacy Policy · Digital Product Terms