Vol. 1 · for engineers

The Claude Code *Field Manual*

A short, friendly guide to working with an AI teammate on real code.

free to share!

MIAN SHAFIQ · SHAFIQ.PK



25 years of code,
one new teammate

Hi, I'm Shafiq.

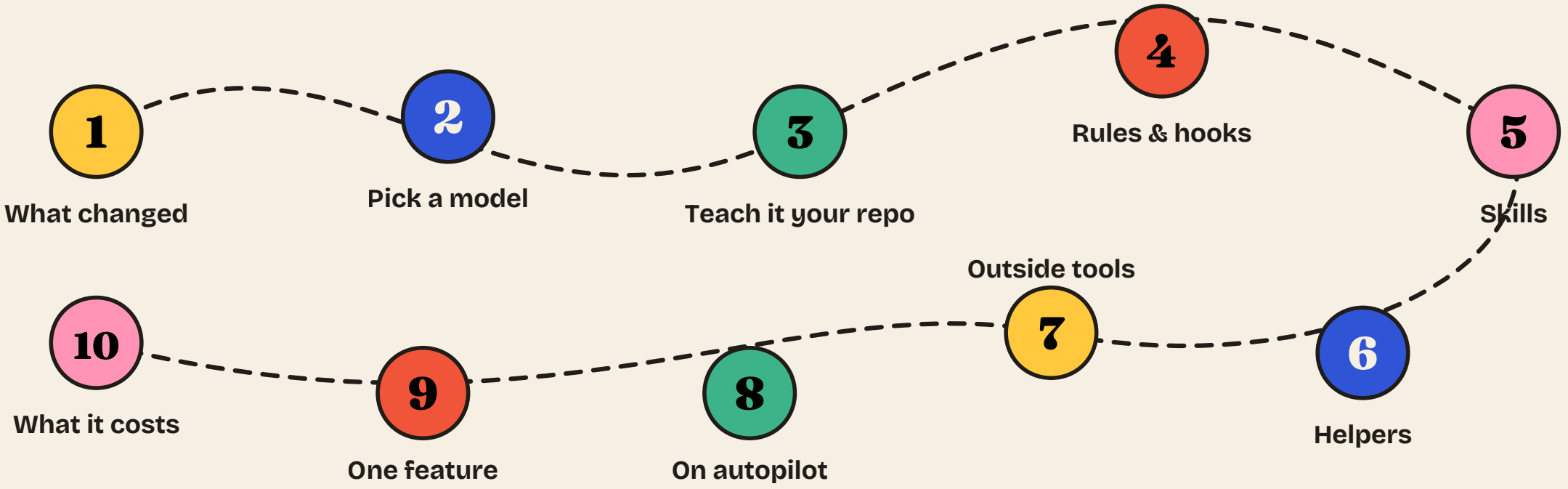
I've shipped software since the days of FTP deploys. Coding agents changed my day faster than anything before them.

This isn't a list of magic prompts. It's how to set up your repo so Claude does good, safe, checkable work.

One idea per slide. About 20 minutes. Then go try it.

The route

Ten stops. The first four are the foundation.



1

a true story...

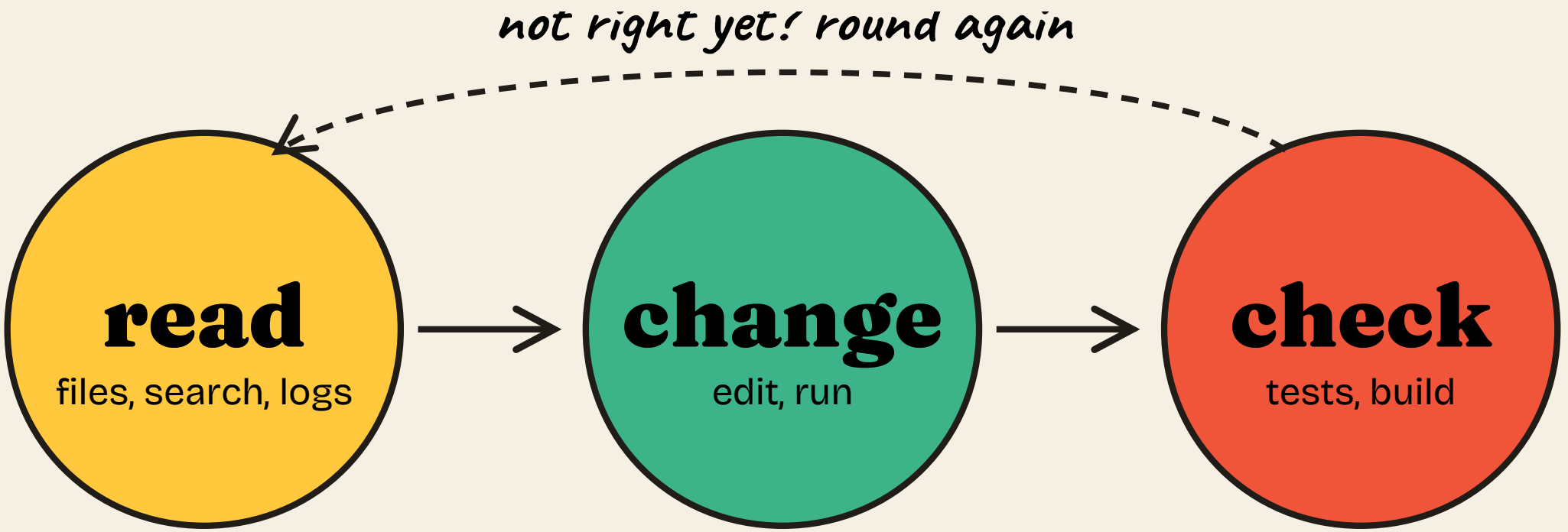
I gave it a whole ticket and a failing test, then went to make tea.

When I came back, the test passed, the diff was small, and the explanation was better than mine.

The unit of work changed from a line of code to a task.

What actually changed

Every session is a *loop*



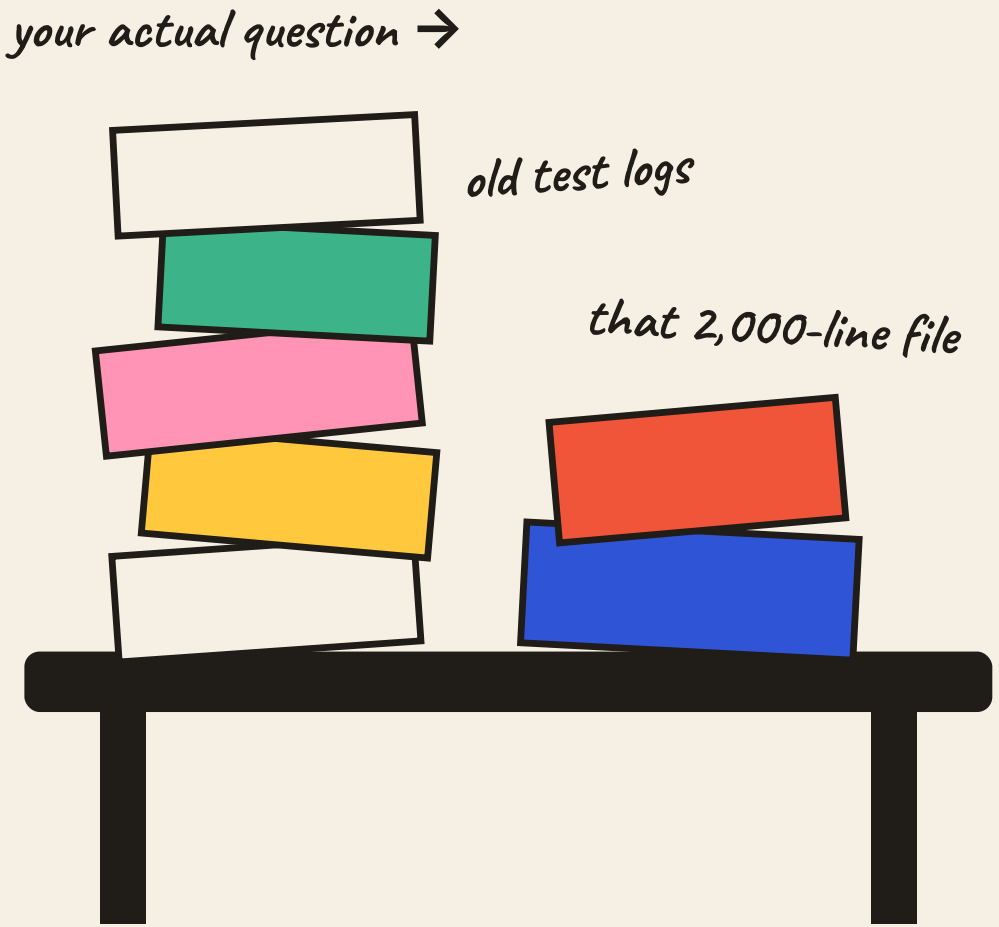
Claude stops when it thinks it's done. **Give it a way to check,** or "looks done" is all it has.

Think of context as a *desk*

Your messages, every file it reads, every bit of test output: it all piles up on one desk.

A messy desk means worse work. Clear it between tasks.

`/clear` fresh start `/context` see the pile



Three habits fix most bad sessions

HABIT 1

Keep the desk clean

Fresh session for each task. Send big reading jobs to a helper.

HABIT 2

Hard rules go in code

A note in a file is a request. A rule or a hook is a guarantee.

HABIT 3

Always give it a check

A test, a build, a screenshot. If it can't check, you'll have to.

2

Meet the team

Four current models. All read huge codebases. They differ in depth, speed and price.

HAIKU 5.5

The quick runner

Searches, sorting, quick lookups.

SONNET 5.5

The steady pro

Clear tasks, CI jobs, helpers.

OPUS 5.5

The senior engineer

Your everyday driver.

FABLE 5.2

The specialist

For the really hard problems.

Pick a model

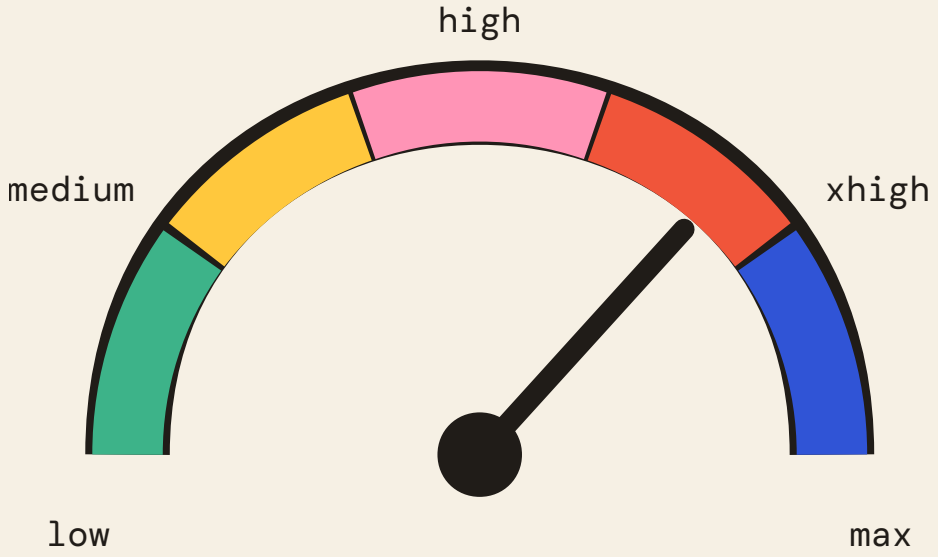
API prices per million tokens (in / out): Haiku from $0.10 / $0.50 · Sonnet $2 / $10 · Opus $4 / $20 · Fable $8 / $40. On a plan, usage comes from your allowance instead.

Turn up the *effort* before you change model

Effort is how hard the model thinks before it acts. If Opus struggles, try it on `high` before calling in Fable.

For routine work, turn effort down first. A cheap model that needs three tries isn't cheap.

/effort high



Watch out: helpers set to `inherit` copy your model. Pin cheap ones to Haiku.

3

lesson learned the hard way...

Our CLAUDE.md hit 400 lines in a month. Claude ignored half of it.

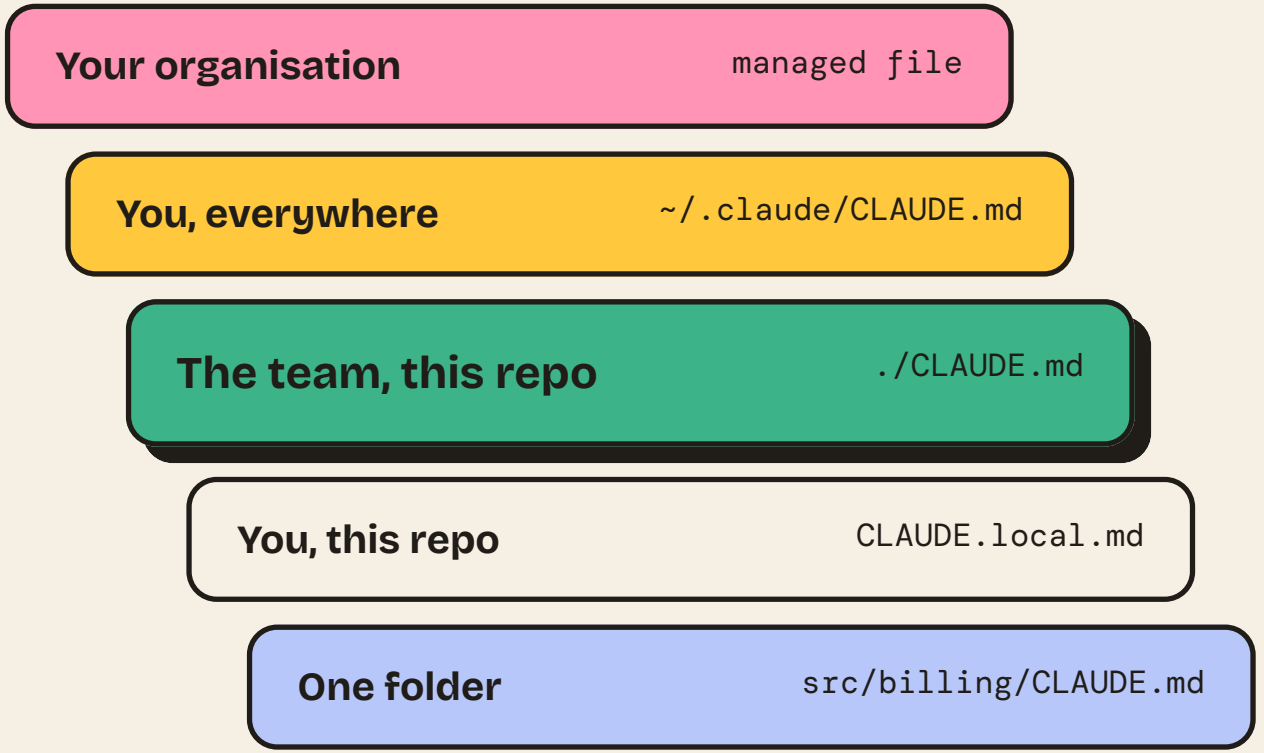
Honestly, so did we. The fix wasn't a better file. It was a **smaller** one, plus a few files in the right places.

Teach it your repo

Instructions come in *layers*

Claude reads them all, general first, specific last. Nothing overrides anything; they stack.

The team file does most of the work. Folder files load only when Claude works there.



Short beats complete

Keep it under 200 lines. For each line, ask:

"Would Claude get this wrong without this line?"

No? Delete it. Claude can read the code.

CLAUDE.md

```
# Payments API  TypeScript · Postgres · pnpm

## Commands
- Test one file: pnpm vitest run path/to/file
- Typecheck: pnpm tsc -b

## House rules
- Money is integer minor units. Never floats.
- Ledger writes only via LedgerService.

## How I like to work
- Bugs: failing test first, then the fix.
- Show me the test output as proof.
```

↑ *commands, quirks, and decisions. That's it.*

Rules that show up only when needed

Put area-specific rules in `.claude/rules/` with a `paths` line. They load only when Claude touches a matching file.

.claude/rules/migrations.md

```
---
paths:
  - "db/migrations/**/*.sql"
---
- Migrations only go forward.
- New column on a busy table?
  Add a default and a backfill plan.
```

`/init`

Drafts a CLAUDE.md. Expect to delete half.

`/context`

Shows what actually loaded.

`/memory`

Claude's own notes on your corrections.

4

on a lending platform...

CLAUDE.md said "never edit migrations". One day it did anyway.

Review caught it. That afternoon the sentence became a hook, and it never happened again.

Writing is advice. Rules and hooks are guarantees.

Rules & hooks

How much freedom? *Shift+Tab to switch*

`plan`
Reads and proposes. Changes nothing.

`default`
Asks before anything risky.

`acceptEdits`
File edits just go through.

`auto` ★
A safety model checks each action.

`dontAsk`
Only pre-approved actions. For scripts.

`bypass`
No checks. Throwaway containers only.



more careful ★ *the default today* *more freedom*

Deny, ask, allow

Claude checks in that order. First match wins, so nothing can allow its way past a deny.

deny → ask → allow

Need it to really hold? Add `/sandbox`.

.claude/settings.json

```
{ "permissions": {
  "allow": ["Bash(pnpm test *)",
            "Bash(git diff *)"],
  "ask":   ["Bash(git push *)"],
  "deny":  ["Read(./.env)",
            "Read(./secrets/**)"]
} }
```

commit this file, so the whole team gets the same rules

Hooks: things that *always* happen

SessionStart	UserPrompt…	PreToolUse	PostToolUse	Stop
add context	screen prompts	**can block it**	**format, lint**	**no finishing while red**

.claude/hooks/protect.sh

```
file=$(jq -r '.tool_input.file_path')
for p in ".env" "db/migrations/"; do
  if [[ "$file" == *"$p"* ]]; then
    echo "Blocked: $p needs a human" >&2
    exit 2
  fi
done
```

`exit 2` blocks the action and tells Claude why, so it changes course.

tip: just ask Claude to write the hook, then read it

5

Explained it twice? Make it a skill.

A skill is a folder with one file, `SKILL.md`. Run it as `/its-name`, or Claude uses it by itself when the task fits.

e.g. When I mentored an intern building small verification services, a `new-service` skill made every one look like it came from the same team.

Skills

A skill worth stealing

The `!` line runs first, so Claude starts with the real commit list instead of hunting for it.

`disable-model-invocation` means only you can start it. Use it for anything with side effects.

.claude/skills/release-notes/SKILL.md

```
---
name: release-notes
description: Release notes from commits
  since the last tag.
disable-model-invocation: true
---
!`git log --pretty='- %s' $(git describe
   --tags --abbrev=0)..HEAD || true`

Group into Added, Improved, Fixed.
Write each line as a benefit to users.
```

/fix-issue

/codebase-map

/review-spec

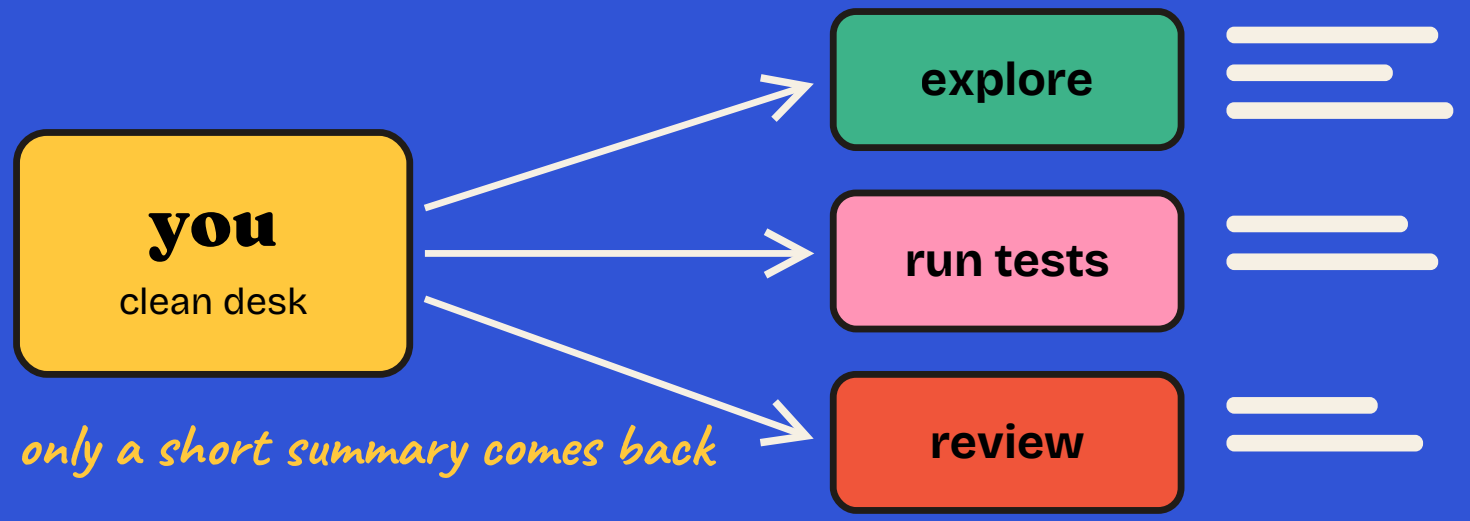
↑ *good first skills. Built-in: /code-review, /verify, /batch*

6

Helpers get their own desk

A **subagent** reads 200 files so you don't have to. Only a short summary comes back to your session.

Helpers & worktrees



Hire a reviewer first

One file in `.claude/agents/`. It reads but can't edit, and it never wrote the code, so it has no reason to defend it.

Call it directly with `@agent-code-reviewer`.

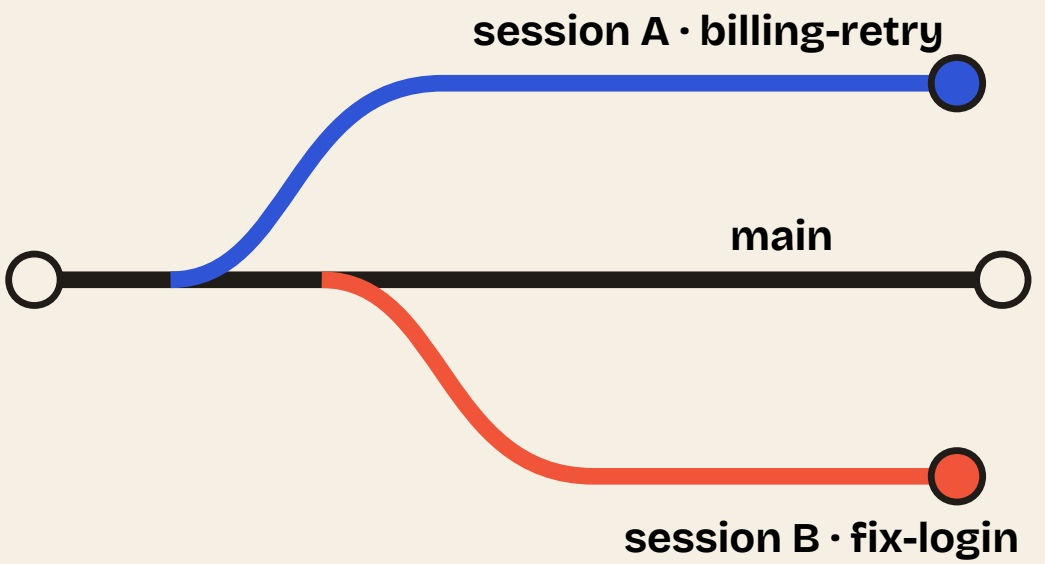
Tell reviewers: **only report what affects correctness.** Otherwise they'll always find "something".

.claude/agents/code-reviewer.md

```
---
name: code-reviewer
description: Reviews a diff. Use
  proactively after changes.
tools: Read, Grep, Glob, Bash
disallowedTools: Write, Edit
model: opus
effort: high
---
Review only the diff. Report bugs,
security issues, missed requirements.
```

Worktrees: two sessions, zero collisions

A worktree is a second checkout on its own branch. Run sessions side by side without stepping on each other's files.



terminal

```
$ claude -w billing-retry       # new worktree + branch
$ claude -w "#1234"             # start from a pull request
$ claude -w billing-retry --resume
```

big rename across the repo? /batch spreads it over many helpers, each in its own worktree

7

CLI first. MCP second.

Claude already knows `gh`, `aws`, `kubectl`, and costs nothing until it uses them. Add an **MCP server** when there's no CLI or you need a login.

true story: our messaging gateway's CPU was maxed out. Claude's guess was "get a bigger server". Once it could read the logs and process stats itself, it found the real cause: browser launch flags and a cache that never reused anything.

Outside tools

```
$ claude mcp add --transport http tracker https://mcp.example.com/mcp
```

8

Add `-p` and it runs on its own

One run, one answer, exit. That tiny flag is how Claude gets into scripts and CI.

terminal

```
$ cat crash.log | claude -p "Root cause? Five lines."
$ claude -p "Fix lint errors in src/" --max-budget-usd 2

/goal all tests in test/payments pass and lint exits 0.
      Stop after 25 turns.
```

On autopilot

`/goal` keeps it working until a separate judge agrees the condition is true. **Put the proof in the condition.**

Tiers of trust

The pipeline I'm building for a small online food shop, in GitHub Actions.

1 · TRIAGE

Haiku labels

every new issue

2 · WORKER

Opus fixes

the clear ones

3 · REVIEWER

Fresh eyes

check every PR

4 · HUMAN

Me

approves the merge

5 · AUTO

Docs only

merge by themselves

every step is cheap. only the human step is slow, and that's on purpose.

9 One feature, start to finish

1 · SPEC

"Interview me, then write SPEC.md." Then close the session.

2 · PLAN

New session in plan mode. `Ctrl+G` to edit the plan.

3 · BUILD

`/goal` with the tests as the finish line.

4 · PROVE

"Show me the test output." Evidence, not "done!".

5 · REVIEW

Reviewer agent on the diff, then open the PR.

Drifting? Press `Esc`. Went wrong? `Esc Esc` rewinds to any earlier prompt. *(still use git!)*

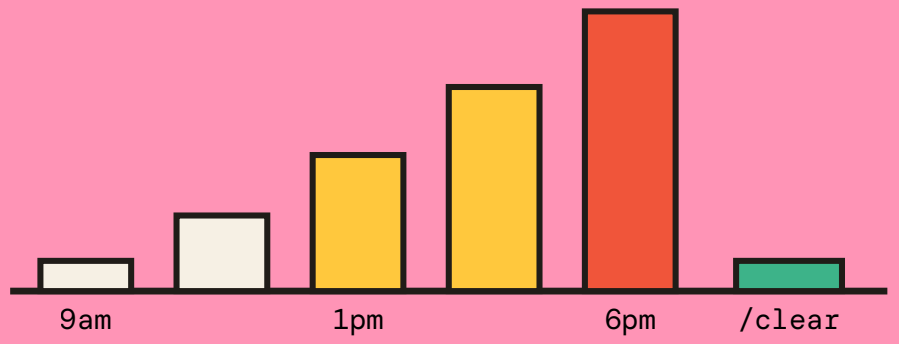
10

About $13 a dev per working day

Anthropic's enterprise average. Most stay under $30 a day.

The surprise: **every message re-sends the whole conversation.** An all-day session makes even tiny questions heavy.

What it costs



I once left a /loop on Opus running all weekend. It found the bug. Priciest bug report ever.

Five cheap habits

`/clear`
between unrelated tasks

`/btw`
side questions that don't stick

HAIKU HELPER
for test runs and logs

SLIM CLAUDE.md
it's sent every time

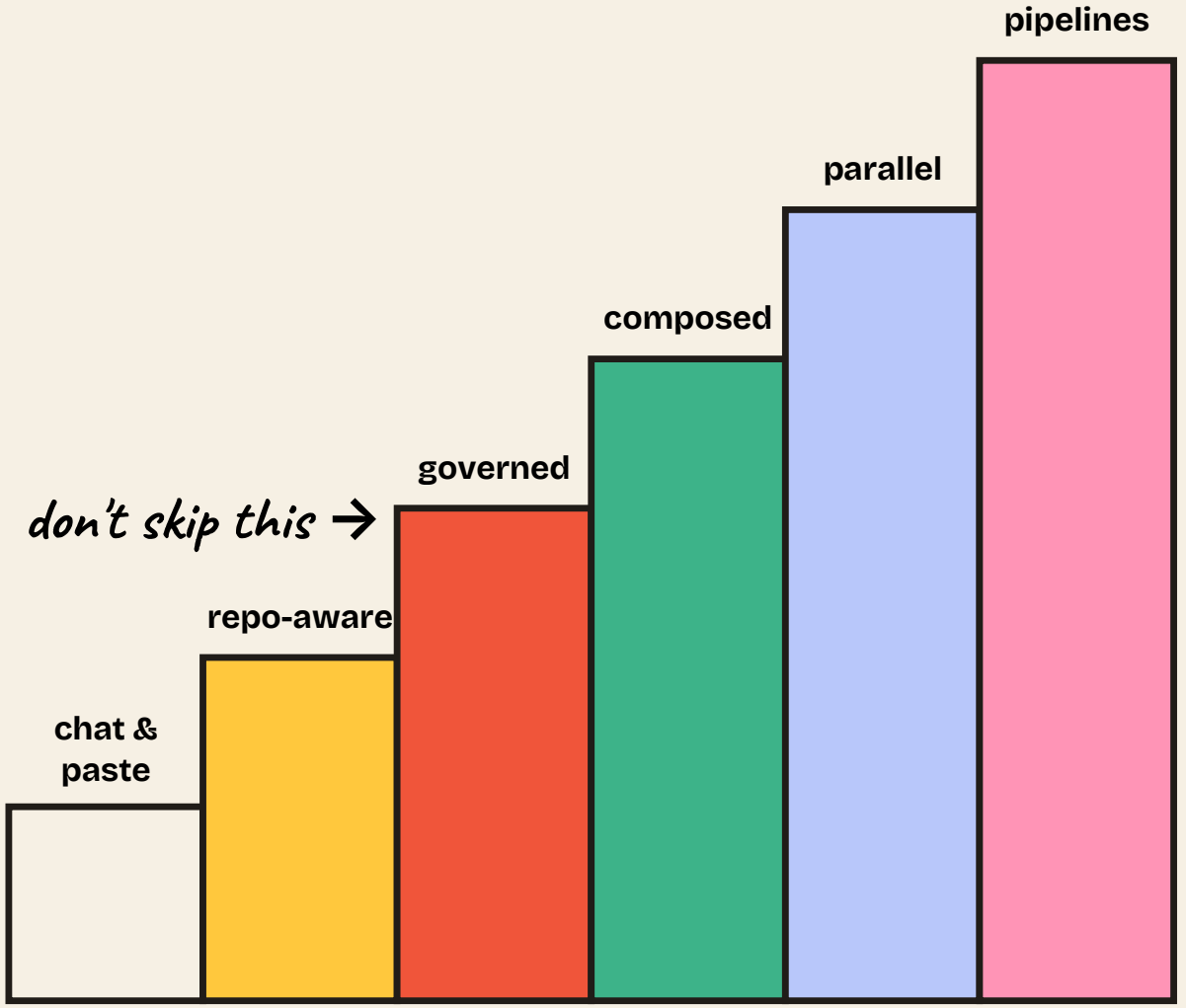
STOP LOOPS
when you walk away

`/usage` shows where it went and suggests fixes. `/insights` shows what keeps going wrong.

Bringing your team along

Skip the mandate. Get one engineer's repo into good shape, then open a PR that adds `.claude/`. The team argues about the rules, and ends up owning them.

Climb one step at a time. Most teams skip governed, the one that makes the rest safe.



Pin this one

EVERY DAY

`/clear` fresh start

`/context` what's loaded

`/usage` where it went

`/btw` side question

`/model` `/effort`

`/rename` `/resume`

GETTING WORK DONE

`/goal` work until true

`/code-review` fresh eyes

`/verify` run the app

`/batch` spread it out

`/loop` repeat on a timer

`/init` draft CLAUDE.md

KEYS & FLAGS

`Shift+Tab` mode

`Esc` stop · `Esc Esc` rewind

`Ctrl+G` edit plan

`Ctrl+B` background

`-p` run once · `-w` worktree

`--max-budget-usd`

FILES

`CLAUDE.md`

`.claude/settings.json`

`.claude/rules/`

`.claude/skills/`

`.claude/agents/`

`.mcp.json`

Now go try it.

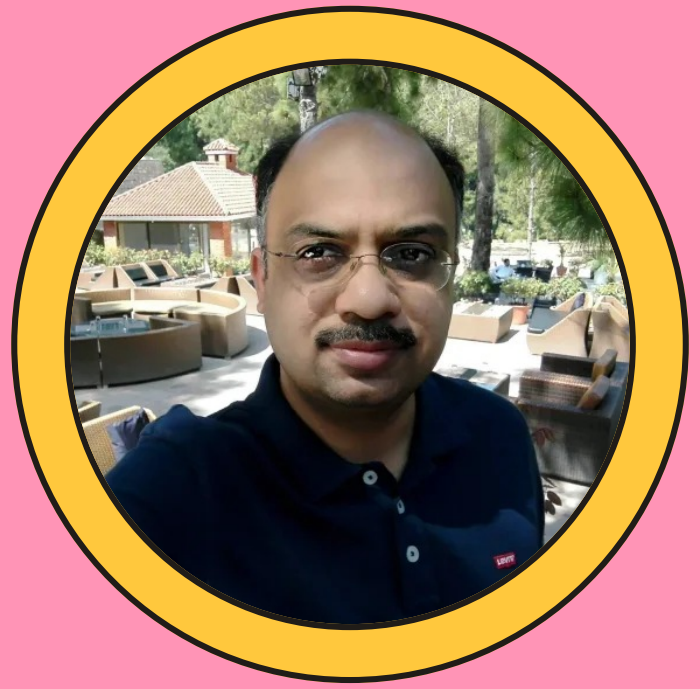
Pick one slide. Do it in your repo today. Tomorrow, pick another.

Mian Shafiq is a technology leader, software engineer and AI practitioner with over 25 years of building software and the teams that ship it, across recruitment, fintech and SaaS.

Next in the Field Manuals: **Claude at Work** · **Claude for the Company** · **Building on Claude**

Free to share. Not affiliated with Anthropic. Checked against Anthropic's docs, November 2026.

shafiq.pk



yes, a real human reads every message

Questions? Comments? *War stories?*

Tell me what worked, what didn't, and what Volume 2 should cover.

EMAIL

talk-to-a-human@shafiq.pk

X / TWITTER

@shafiq

LINKEDIN

in/phpgurru

WEB

shafiq.pk