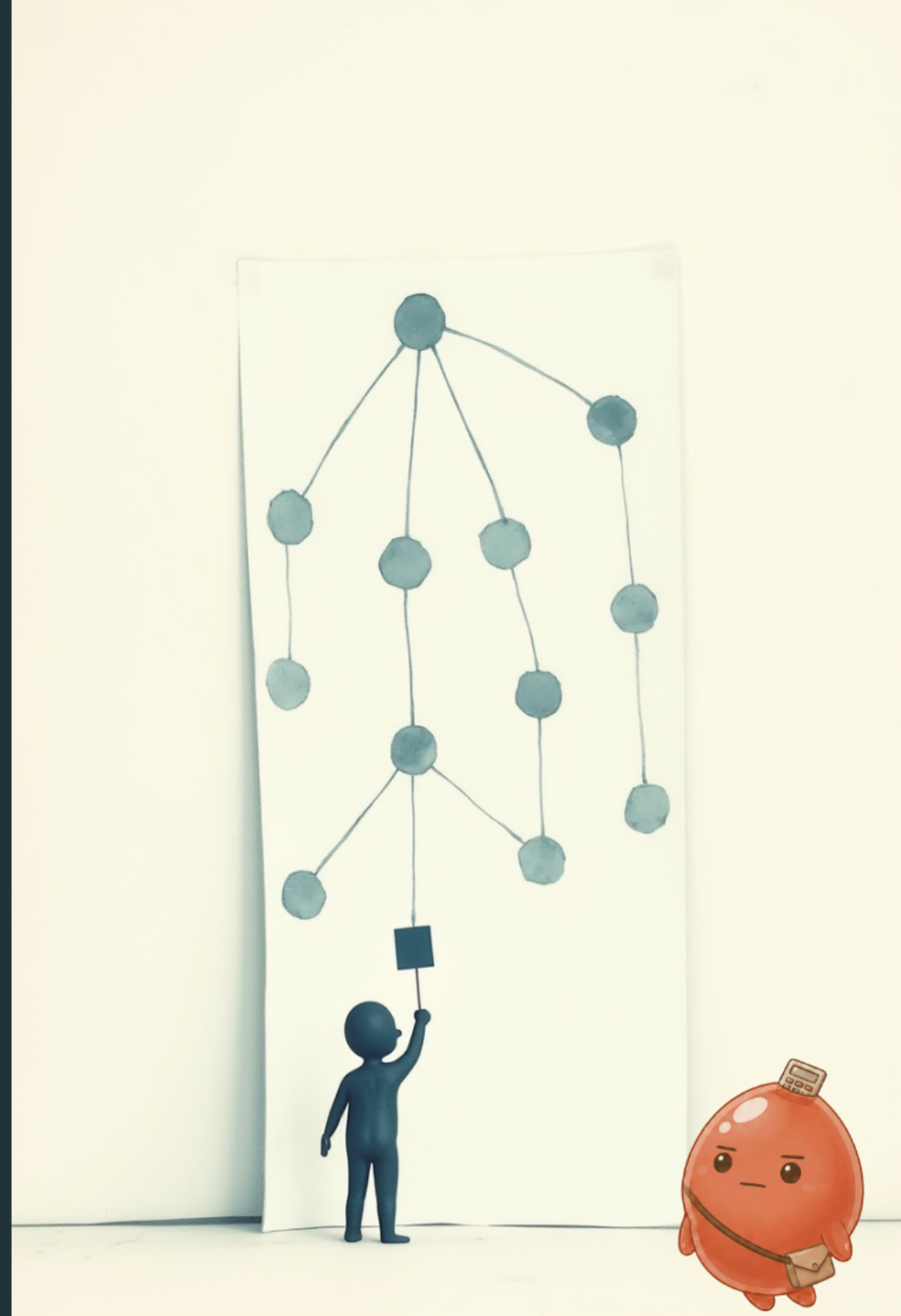


どこに何を置くか、 一枚の地図にする

言語化した設計を、Claude Code の環境に配置する回



増えつつける「棚」に、名札を貼っていく

hooks・skills・agents・rules。仕組みが増えるほど「どこに置くか」があいまいになる。

プロジェクトの一番上から順に降りて、棚ごとに「ここは何を置く場所か」の名札を貼っていく。



① 個人メモが、チーム共有に混ざってしまう

自分用の開発URLやデバッグ設定が、チーム共有のルールファイルに乗って、全員に配られてしまう。

たとえ：会議室のホワイトボード（全員で見る）と、自分の手帳（自分だけ）。

共有 = CLAUDE.md（Gitで管理）／個人 = CLAUDE.local.md（リポジトリに含めない）。

どちらもルート直下。CLAUDE.md は長すぎると読まれにくいので、ふくらんだら分割。



① .mcp.json は、置き場所を間違えやすい

外部ツール接続の定義ファイル。 .claude/ の中に入れると、つながらない。

たとえ：郵便受けは玄関の外、と決まっている。 Claude Code もこのファイルを決まった場所では探さない。

置き場所は .claude/ の中ではなく、プロジェクトのルート直下。ここだけ、と覚える。



② なぜ、設定ファイルが別に要するのか

お願いごとの文章と、オン／オフのスイッチは種類が違う。混ぜると読みにくく、AIも迷う。

CLAUDE.md = 同僚への申し送りメモ（意図をくみ取る）。

settings.json = 機械のスイッチ盤（オンかオフか、迷う余地なし）。

スイッチ盤 = settings.json。JSON という形式で、きっちり書く。

自分だけの設定は settings.local.json（CLAUDE.local.md と同じ考え方）。



② 危険なコマンドを、勝手に実行されたら困る

全部消すようなコマンドは止めたい。でも安全なコマンドで毎回許可を聞かれるのも面倒。

たとえ：入っていい部屋（許可リスト）と、鍵をかける部屋（禁止リスト）。

permissions に、allow（ふだん使うもの）／ask（そのつど確認）／deny（危険なもの）を書き分ける。



③ 決まった作業を、毎回手で指示したくない

ツールを使う前後に、危険チェックやテストを、自動でやってほしい。

たとえば：家を出る前に自動で戸締まり確認（前）／料理のあと自動で皿洗い（後）。

使う直前 = PreToolUse（危険なコマンドを止める）／直後 = PostToolUse（整形・テストを走らせる）。

スクリプトの置き場 = `.claude/hooks/`

ここだけ他の棚と違う。置くだけでは動かない。

`settings.json` の `hooks` に「いつ・どれを走らせるか」を登録して、はじめて動く。



③ 開始時と、応答の終わりにも差し込める

SessionStart = 開始時にブランチ名などを自動で渡す。

Stop = 応答の終わり。落ちたテストがあれば、作業を続けさせる。

第3回で「呼べば動く」まで保留した検証の自動発動は、
この hooks に登録して初めて実現できる。



④ 気に入ったやり方を、毎回説明したくない

手順と、見本と、道具を、ひとまとめにして呼び出したい。

たとえば：レシピカードと材料と道具を、1つの箱にまとめておく。

この箱 = skills/ - 1ディレクトリ = 1手順書。

中に SKILL.md（手順） + examples/（見本） + scripts/（実行用）。



⑤ 決まった役割は、丸ごと任せたい

調べもの・レビューなどを、今の会話をそのままに、専門の担当へ。終わったら結果だけ受け取る。

たとえば：別室の専門スタッフに、必要な資料だけ渡して任せる。

agents/ — 1ファイル = 1エージェント（tools・model を絞れる）。

スキル=いまの Claude に読ませる手順書／エージェント=別コンテキストで動く独立した担当。



⑥ CLAUDE.md が、長くなりすぎた

API の決まり、テストの決まり…全部を1ファイルに書くと重く、関係ないときも読まれる。

たとえば：分厚いマニュアルをテーマごとの薄い冊子に分け、必要な冊子だけ開く。

rules/ に用途別で分割。paths を指定すると、その場所のファイルを触るときだけ効く（コンテキストと費用の節約）。



⑦⑧⑨⑩ その周りの、軽い仕組み

output-styles/ — 口調や答えの形そのものを入れ替える

commands/ — 短いスラッシュコマンドを1ファイルずつ（今はスキルが主役）

plugins/ — 配られたプラグインが入る場所（自分で配るときは専用フォルダを別に作る）

statusLine — フォルダではなく settings.json の項目。画面下部に常駐表示

「フォルダではないもの」も混ざる。statusLine は settings.json 側。



⑪ ~/.claude/ — 全プロジェクト共通の設定

- ~/.claude/settings.json — 全体のベースライン設定
- ~/.claude/CLAUDE.md — 自分全体のルール
- ~/.claude/projects/ — プロジェクトごとの Auto Memory

プロジェクトの設定が優先。ユーザー設定はベースライン、プロジェクトで上書きできる。



⑫ 設定ファイルは、全部自分で書かなくていい

hooks・skills・agents・rules...全部、自分でコードを書くのは大変そう。

たとえ：設計士と大工。設計するのは自分、形にするのはAI。

自分が決めるのは「何を禁止するか／どのタイミングで何を走らせるか／なぜそう決めたか」だけ。

中身のコードは Claude Code に書いてもらえる（permissions・hooks・statusLine・SKILL.md も）。

書けないことは気にしなくていい。決めることさえできれば、形にするのは任せられる。



迷ったら、設定の優先順位

強い順に：組織 → CLI → local → project → user
効かないときは、この順のどこかで上書きされている。

- ☐ 自分の .claude/ が地図のどこに当たるか確認
- ☐ hooks/ に置きそうな検証を1つ見つけた
- ☐ 設定の優先順位を理解した

次回「次の扉：自走」。

発動のきっかけが外の出来事変わったとき。

