Task Decomposition for Agents

A cheat sheet. *When to give Agent the whole job, when to break it up. Three dimensions, one question: how big a swing is too big?*

01 Scope

How much code is touched.

One-shot if the change fits in one file or one mental model.

Split if the change spans 5+ files, multiple modules, or unfamiliar boundaries.

02 Reversibility

How hard is it to undo if wrong.

One-shot if git checkpoint covers it. Editor diff is the rollback.

Split if external state is involved: DB migration, CI config, deployed service, third-party API.

03 Specifiability

How precisely can you describe "done."

One-shot if you can write a failing test or a precise diff in <2 minutes.

Split if the goal is fuzzy ("make this faster," "clean it up") — Plan first, then split.

Blast-radius lookup

First read of the task. Match the row, pick the move.

Local & reversible	git diff is the safety net. Let Agent run.
Adds new file, no other touches	low blast radius. One-shot.
Edits one file, behavior change	split: write the test first, then the change.
Edits 5+ files, same pattern	split: do one as a worked example, then batch.
Touches public APIs	split: design first (Plan), then implement, then update callers.
Migration / rename across repo	split by directory, verify after each batch.
Touches infra, secrets, CI	human writes; Agent reviews. Don't reverse.

Splitting strategies

When you've decided to split, pick the cleavage line.

By layer. UI → controller → service → DB. Land one layer at a time. Each ships independently.

By file batch. Repetitive migrations. Do one as a worked example. Then "do the same for these 12 files."

By contract. Define the new interface first; commit it; then implement on each side. Keeps both sides green.

By guardrail. Add the test, fail it; ship the test alone; then implement. Forces a clean signal.

A Anti-pattern: the mega-prompt

"Refactor the auth system, add MFA, migrate to JWT, and update all the tests."

Why it fails: Agent commits early, runs out of context mid-job, or pattern-matches one piece against another. The diff becomes unreviewable.

Fix: split by layer or by guardrail. Land each piece. Re-prompt with fresh context.

B Anti-pattern: over-splitting

"Step 1: rename one variable. Step 2: rename another."

Why it fails: you spend more time coordinating than coding. Trivial mechanical edits should batch — that's what tools are for.

Fix: if every split is reversible and inside one file, merge them. Three similar lines is better than three prompts.

C Anti-pattern: hidden coupling

"Just update the response shape. The frontend is fine."

Why it fails: "fine" is unverified. Splits across system boundaries need a contract — not a guess.

Fix: define the contract first (TypeScript types, OpenAPI, schema). Implement on both sides against the same artifact.

The 30-second test. Before you hit Enter: can you describe the failure mode in one sentence? If yes, one-shot it. If no, you don't know what "done" looks like — Plan first.