

Parallel Design Variants

Fork the same scaffold into five subagents with five different art directions, then pick one or graft the best parts into a sixth.

01 When to use this

Use this once the dashboard structure is settled and the bare scaffold works. It replaces the loop where you iterate one design five times in a row. Reach for it any time you have branches that do not touch each other.

02 The prompt

The scaffold at `<SCAFFOLD_PATH>` is structurally correct. Do not change the structure, the data, or the page list.

Spawn `<NUMBER>` subagents in parallel. Each one produces a complete styled version of this same scaffold in its own directory, working from its own art direction brief. They do not read each other's work and they do not touch each other's files.

Art directions, one per agent:

1. Editorial - magazine typography, generous whitespace, strong headline hierarchy.
2. Terminal - monospace, dark, dense, high contrast, minimal chrome.
3. Swiss - grid discipline, restrained palette, precise alignment, no decoration.
4. Dense ops - maximum information per screen, small type, built for an operator who reads it daily.
5. Warm print - paper tones, soft contrast, printed report feel.

One shared constraint every variant must satisfy: `<SHARED_CONSTRAINT>`, for example every variant must surface the same seven anomalies above the fold. A variant that fails this constraint is disqualified no matter how it looks.

When all agents are done, show me each variant as a screenshot plus its path, side by side, in one message.

After I pick, I may ask you to graft parts of several variants into a sixth build. Keep every variant on disk until I say otherwise.

03 What it is actually doing

Iterating one design five times means waiting five times. If each pass takes five minutes you have spent twenty plus minutes staring at a progress bar, and you only ever saw one lineage of ideas.

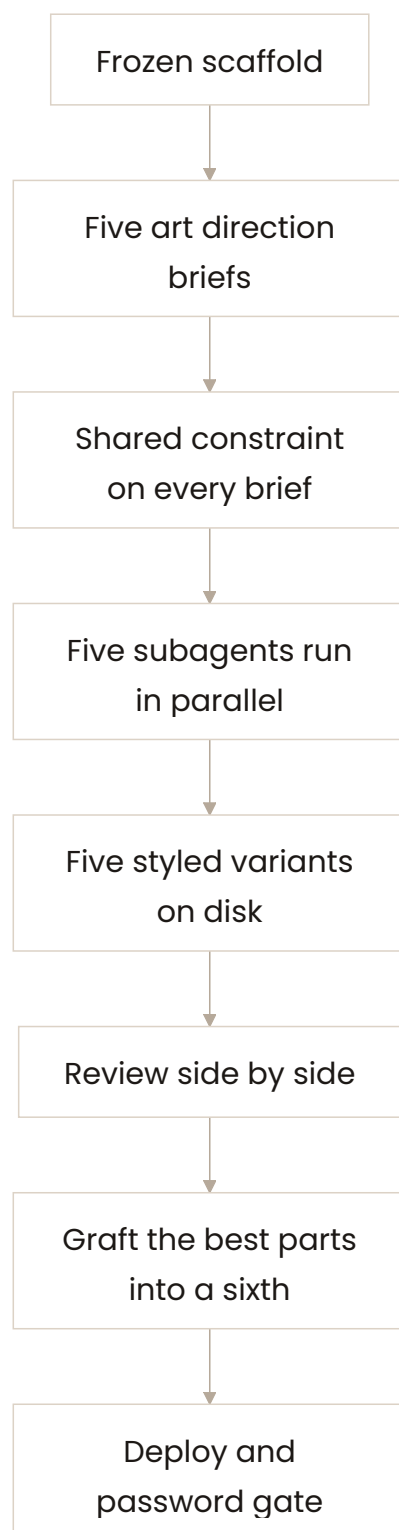
Running five agents at once takes about five minutes for all of them, plus one merge pass at the end. You also see five genuinely different answers instead of five versions of your first idea, which matters because design is a search problem and you want to search wide.

The rule for when this is safe is simple. A task is parallelizable when the branches are independent and do not affect each other. Five design variants of the same scaffold qualify, because each writes to its own directory. Sequential edits to one file do not qualify, because the second write wins and quietly destroys the first.

There is a real price. You burn usage limits several times over, and each branch fails more often because nothing is watching it while it runs. That failure rate is acceptable here for one reason: you only need one of the five to be good.

The last piece is the shared constraint. Give every brief one requirement they all have to meet, so the variants stay comparable. Nick required every variant to surface the same seven anomalies. Without that, you are comparing five different dashboards rather than five treatments of one.

04 How it flows



05 Step by step

01 Freeze the scaffold first

Structure, data, and page list are settled before you fork. If the branches can change structure, you cannot compare their designs.

02 Check the branches are independent

Each agent writes to its own directory and never reads another's. If two branches would edit the same file, this is the wrong tool.

03 Write genuinely different briefs

Editorial, terminal, Swiss, dense ops, warm print. Distinct art directions, not five adjectives that mean the same thing. The point is coverage of the solution space.

04 Add one shared constraint

Give every brief the same hard requirement, like surfacing the same seven anomalies. This keeps the variants comparable and disqualifies pretty but useless output.

05 Accept the failure rate

Nobody is watching each branch, so some will come back broken. That is fine. You need one good one, not five.

06 Review them side by side

Ask for all variants in one message with screenshots and paths. Judging them together is much faster and more honest than one at a time.

07 Graft into a sixth

You rarely love one whole variant. Take the header from one and the table treatment from another and have Claude build a sixth from the parts.

08

Host and gate it

A static host MCP connector lets Claude deploy and password-protect the build directly. Gate per client with a URL parameter plus a per-client password.

06 **Words explained****Subagent**

A separate Claude working its own task with its own context. Several can run at the same time on the same project.

Parallelizable

Safe to run at the same time, because the pieces do not depend on or overwrite each other.

Art direction brief

A short description of the visual world a design should live in, given as a starting point rather than a spec.

Wall clock time

How long you actually wait, as opposed to how much total work got done. Parallel work cuts wall clock, not total work.

Usage limits

The cap on how much you can run in a window. Five agents at once spend it about five times faster than one.

Graft

Combining the best pieces of several finished versions into one new build.

07 Watch out for

-
- Parallel agents on the same file destroy each other's work. The second write wins. Only fork independent branches.
-
- Five vague briefs produce five nearly identical designs. Make the art directions genuinely different.
-
- Without a shared constraint you get five incomparable dashboards instead of five treatments of one.
-
- Nothing monitors a branch while it runs, so expect some to come back broken. Budget for that instead of debugging each one.
-
- This burns usage limits several times faster than working serially. Use it for breadth, not for routine edits.
-

08 How it connects

Dashboard Structure Derivation

That note produces the frozen scaffold. This one is worthless without it, because there is nothing stable to fork.

Design Tuner App

Both notes solve the same problem, which is that describing a visual change in words is slow. One exposes sliders, the other spawns options.

Marketing Dashboard Build Pipeline

The framework note recording the five art directions, the tradeoff table, and the hosting step.

ICP Self-Validating Scrape

Both are ways to spend compute instead of your attention. One iterates against a threshold, the other searches wide in one shot.

Credit. Framework and approach by Nick Saraev, from the parallel design variant segment of the dashboard build in his Zero to One course on Claude Code for marketing. Written up for the Kape vault by BrewedOps.