Building With an Agent

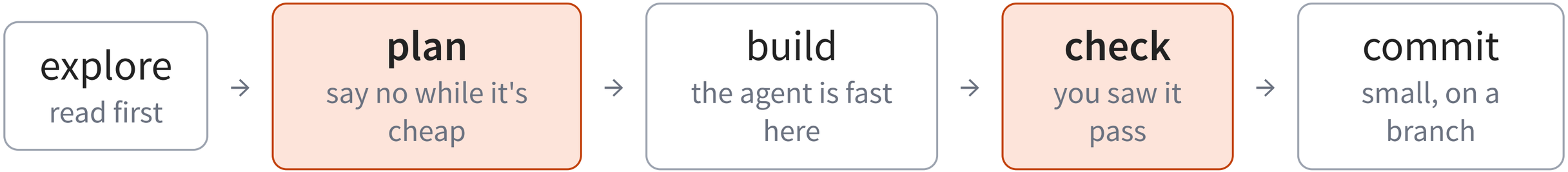
The loop, the checks, and the tools around it

E190BO · AI-Amplified Prototyping

Five big ideas

1. **Explore, plan, build, check, commit.**
2. **Give it a way to check its own work.**
3. **Git is your safety net and your review surface.**
4. **Teach it about your project.**
5. **Same architecture, new app: read hardest where they differ.**

Every slice goes round the same loop



The two orange steps are slow, and they're the ones people skip. *"This should now work"* is a claim, not a check.

A check turns a claim into a loop

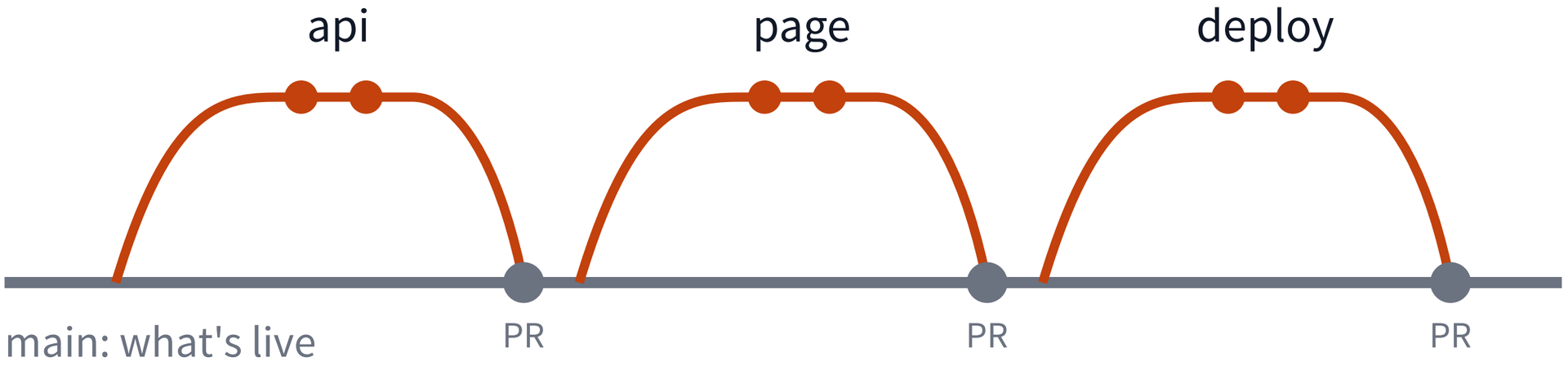
without a check
writes code, says "done"

run the tests → read the failure → fix → ↺ until **they pass**

> Build the API until `pytest -m contract` passes. Don’t touch the tests.

The shortcut to green is to **edit the test**, so protect the tests. And no test sees the page: there, you’re the check.

One slice, one branch, one pull request



Commit each time something works. The agent can open the PR with `gh`; merging to `main` deploys.

The reading is yours: every line of the diff

Does it match?

the contract, and the stack in `CLAUDE.md`

Could I explain it?

if not, ask the agent, or take it out

What didn't I ask for?

agents love tidying up

Usual suspects

a rule only on the page, `innerHTML`, a hardcoded host, a secret, a changed test

Leave a review comment on a line, then merge.

`CLAUDE.md`, and four ways to extend it

CLAUDE.md

a fact it needs *every session*

hook

a rule that holds *every time*

skill

a procedure *when relevant, or /name*

subagent

a side job *in its own context*

MCP server

a new ability *like seeing a page*

All four are files you commit in `.claude/` or `.mcp.json`, so every session gets them. Lab 7 asks for one: pick the one that fixes something that went wrong.

A hook is a rule the agent can’t skip



`CLAUDE.md` *asks* it not to edit the tests; the hook *makes sure*. The Lab 7 reference has this one, in `.claude/settings.json` and `.claude/hooks/`.

Same architecture, new app

`Idea`	`Vote`	`vote_count`	`GET /ideas`
→ `Dish`	→ `Rating`, 1 to 5 stars	→ `rating_count`, `avg_rating`	→ `GET /dishes`

Pin the stack you know in `CLAUDE.md`, because **you can only check what you can read**. The concept map in your `README.md` lines up every piece like this.

Where the old answer is wrong, read hardest

⚠️ `AVG` of nothing

is `NULL`, not 0: an unrated dish isn't a zero-star dish

⚠️ Sorting `NULL`

SQLite and Postgres disagree: green here, wrong on your live site

⚠️ Stars 1 to 5

checked by the API, because `curl` doesn't use your page

The agent has seen a thousand vote counters, and it will copy the old answer.

What Lab 7 asks for

`CLAUDE.md` first

committed before any app code

Slices as PRs

each one read, line by line

The tests

the finish line, untouched

One extension

a hook, skill, subagent or MCP server

One feature

of your own

A concept map

in your `README.md`

At the check-in you explain your app, including the code your agent wrote, **without AI**.