Effective Workflows for Building with Agentic AI

Worktrees, previews, MCP servers and skills

E190BO · AI-Amplified Prototyping

Three parts, six ideas

1 · Workflow

Your reading is the limit, not the agent's speed.

One task, one worktree, one pull request.

Every pull request gets a live URL.

2 · Customizing the harness

MCP servers give it abilities.

Skills give it know-how.

3 · Setup and tips

When something goes wrong twice, write it down.

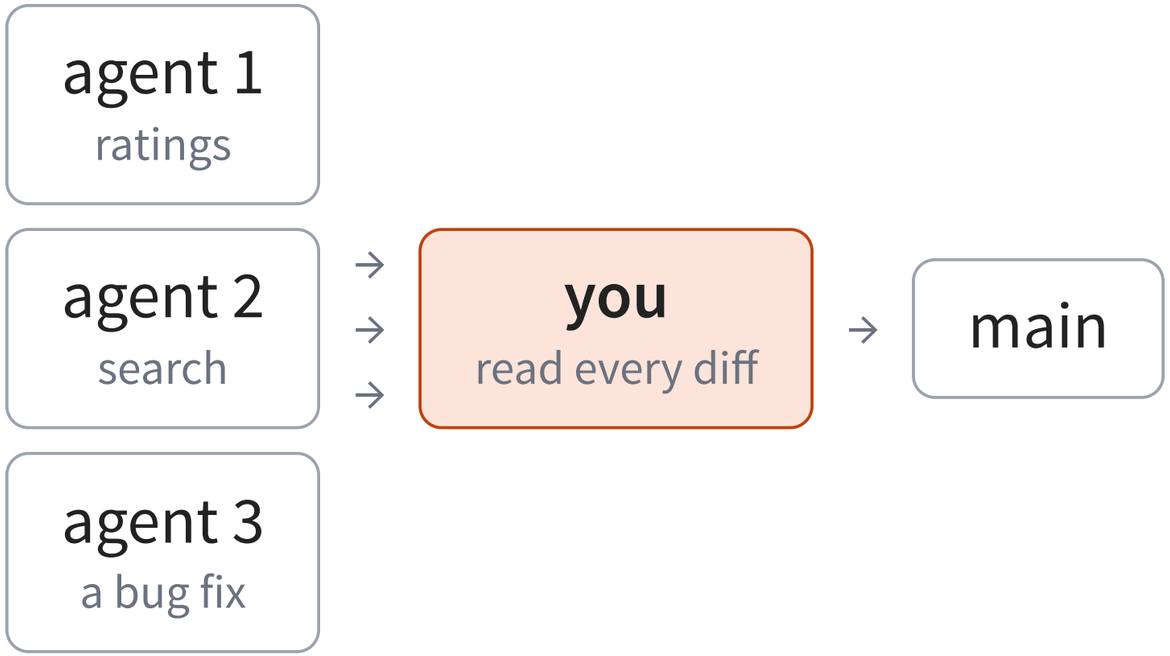
Set the repo up once, for the whole team.

Workflow

1 · Workflow
2 · Customizing the harness
3 · Setup and tips

Run several agents at once, and still know what each one did.

Agents write fast; you read at human speed



Run as many agents as you can **review**, not as many as you can start.

WHY PARALLEL

an agent takes minutes per task; read one diff while the next one builds

USE IT WHEN

the tasks are independent, and you can say exactly what each should do

SKIP IT WHEN

you're still deciding what to build, or two tasks touch the same files

A worktree: one repo, several working folders

one repo

one shared history, one remote

↙ ↓ ↘

the repo itself

on `main`

`ratings/`

in `.claude/worktrees/`, on `worktree-ratings`

`search/`

in `.claude/worktrees/`, on `worktree-search`

```
claude -w ratings        # in one terminal
claude -w search         # in another
```

WHY

two agents in one folder edit the same files, and one branch can't hold two tasks

USE IT WHEN

two agents work at once, on different parts of the app

SKIP IT WHEN

it's one task at a time: a plain branch is enough

Apps that run several agents at once

a task → **a workspace** (a worktree and a branch) → set up (copy `.env`, install) → the agent works (on its own port) → you review (the diff, then a PR)

Claude Code

`claude -w`, or parallel sessions in its desktop app

Codex app

OpenAI's, for Codex agents

Conductor

Mac; Claude Code, Codex and others

Emdash

open source; Mac, Windows, Linux

Orca

open source; Mac, Windows, Linux

Claude Squad

open source; in a terminal, even a codespace

WHY

by hand, every task is a worktree, a `.env`, an install and a free port

USE IT WHEN

you run two or more agents most days

SKIP IT WHEN

you work one task at a time: `claude -w` or a branch is enough

Every task goes round the same loop



Write the plan down in the pull request (PR), or an issue, not only in the chat.

WHY WRITE IT DOWN

the plan outlives the chat: a teammate, or the next session, can read it

WHY A PR

someone reads it before `main` changes, and it gets a preview

WHY A NEW SESSION

a long one fills with old context, about code that's since changed

Every pull request gets a live URL

open a PR → **Railway builds a copy** your app and its own database → a bot comments the URL on the PR → merge or close the copy is deleted

Railway: Project Settings → Environments → **Enable PR Environments**, and invite every teammate to the project.

WHY

no test sees the page, and your codespace runs SQLite, not Railway's Postgres

USE IT WHEN

a change is something a person sees, or it touches the database or the deploy

SKIP IT WHEN

the change is a README; and close stale PRs, since each copy uses credit

Cloudflare Workers does the same: a preview URL on every PR, and one per branch.

Three checks before you merge

The diff

you read every line, in Files changed

catches: code you didn't ask for, a secret, a changed test

CI

the tests run on GitHub, on every push

catches: the test nobody ran before pushing

The preview

click through it, on your phone too

catches: a broken page, a bug only Postgres has

The agent saying “done” is none of these.

Customizing the harness

Give the agent abilities it doesn't have, and know-how about your project.

MCP: give the agent an ability it doesn’t have

Claude Code → asks **an MCP server** (a list of tools, each with a description) → calls Cloudflare (its docs, your account)

```
claude mcp add --transport http cloudflare-docs \
  https://docs.mcp.cloudflare.com/mcp
```

WHY

the agent knows its training data and its terminal; an MCP server connects it to something live

USE IT WHEN

there's no command-line tool for it, or its docs changed after the model was trained

SKIP IT WHEN

a command-line tool exists: `gh`, `railway`, `wrangler`

A skill: a procedure it loads when it needs it

every turn
one line: its description

→ you ask:
"deploy to Railway"

now it loads
the whole `SKILL.md`

→

it follows the steps
with the `railway` CLI

```
railway skills install --agent claude-code      # adds use-railway
```

WHY

`CLAUDE.md` is loaded every session; a skill's steps load only when they're needed

USE IT WHEN

you've explained the same multi-step job twice: deploy, seed the database, release

SKIP IT WHEN

it's a fact every session needs (`CLAUDE.md`), or a rule that must hold (a hook)

MCP server, skill, or a command it already knows?

MCP server

it *can't reach* the thing: a browser, a service with no command-line tool

Skill

it can reach it, but doesn't know *your steps*

A tool it knows

`git`, `gh`, `railway`: just say to use it

Often the answer is the third, plus a skill. Each thing you add is one more thing it has to choose between.

An MCP server acts as you

Cloudflare's docs server

reads public docs; no sign-in

Cloudflare's API server

signs in as you, and can change or delete what you own

Grant the **fewest permissions** that do the job, and only add servers you trust.
Whatever a tool reads (a web page, an issue, a log) is text the model may **follow**.

Setup and tips

Set up the repo once, so every agent on the team starts from the same place.

Twice is a pattern: write it down

wrong once
fix it, move on

→

wrong twice
write it down

a fact

→ `CLAUDE.md`
loaded every session

a procedure

→ a skill
loaded when it's needed

a rule it must never break

→ a hook
enforced every time

Commit it, so every teammate’s agent stops making the same mistake.

Set up your project repo

`CLAUDE.md`

the stack, the commands, the rules

Issues and PRs

everything reaches `main` through a reviewed PR

CI

the tests run on every push

PR environments

on Railway, every teammate invited

Worktree setup

`.worktreeinclude`, or a setup script

One extension

a skill or MCP server your team agrees on

Then every agent, in every worktree, starts from the same place.