

One human, many agents.

Keep the ambition. Make the project operable for its next actor.

Preserve the history. Make corrections change behavior.
Give the next actor a working view.

Eight practices to try in your own project

- 1 Make corrections take effect.** Change the form, tool, check, or workflow the next actor uses. Preserve the reason; verify the next use.
- 2 Keep history; design the working view.** Separate what happened from what to do now. Retain provenance, current obligations, and recovery requirements.
- 3 Give each assignment a contract.** Name the outcome, inputs, territory, authority, stop rule, and done-when. Match the record to the consequence.
- 4 Own mutable resources.** Files, databases, ports, credentials, and targets each need clear ownership. Worktrees isolate only part of the system.
- 5 Use human judgment deliberately.** Bound routine work with real permissions. Route consequential choices with evidence; avoid bottlenecks and rubber stamps.
- 6 Test a cold pickup.** Give a fresh actor only the durable record. Find where they must guess about state, authority, evidence, or the next action.
- 7 Check the check.** Bind evidence to the exact candidate. Confirm that the check detects the failure it claims to exclude, using an isolated test.
- 8 Review the operating burden.** Repair stale views and unused rules. Limit dispatch to review capacity; retire deliberately and preserve the history.

Plan for the obligations that accumulate

One month

Finish and hand over.
Bound the outcome, version the inputs, repeat the checks, and know how to recover changed state.

Six months

Continue through change.
Preserve decisions and dependencies; budget review, onboarding, drift checks, and maintenance.

One year

Survive replacement.
Own services, access, migrations, restore objectives, support obligations, and exit paths.

Consequences can pull any control forward. Plan near-term tasks in detail; plan distant outcomes, interfaces, assumptions, and revisit triggers.

One human, many agents.

Pick one habit and one guardrail. Try them on a task you already care about.

The six-field task card

Outcome	What useful result should exist?
Inputs	Which revision, facts, dependencies, and checks apply?
Territory	Which files, environments, and shared resources may change?
Authority	Which actions are allowed, on which target, under what limits?
Stop / budget	When should the worker stop, report, or ask for a decision?
Done when	What evidence establishes completion, and who accepts it?

For a deployment: record intent before apply; observe before acceptance.
If the outcome is unknown, inspect the target before retrying the operation.

Small practices that make the next session easier

One status view	Show task/revision, what changed, running jobs, and decisions needed. Add freshness and source links.
One inbox	Land new inputs where startup will find them. Give temporary reminders a review date; retain durable obligations.
A quiet question queue	Batch ordinary questions through one reporter. Define urgent conditions that must interrupt immediately.
Disposable test state	Give each worker its own files and test database. Check inherited settings before a test touches storage.
A note to tomorrow-you	Changed + evidence; still uncertain; next action; running jobs. Update the current-work indicator.

Choose the smallest version that helps

Start: one task card, one isolated workspace, one evidence folder, one handoff.
Add services for a job: source/review, task state, execution, evidence/restore, access/observation. Name who maintains and recovers each one.
Try a cold pickup: can a fresh actor identify state, authority, evidence, and the next action? Copy the short records in quick-start.md; use templates.md for higher-consequence work.

Technical references

- [Git worktrees](#)
- [PostgreSQL locking](#)
- [Temporal idempotency](#)
- [OpenTelemetry Collector](#)