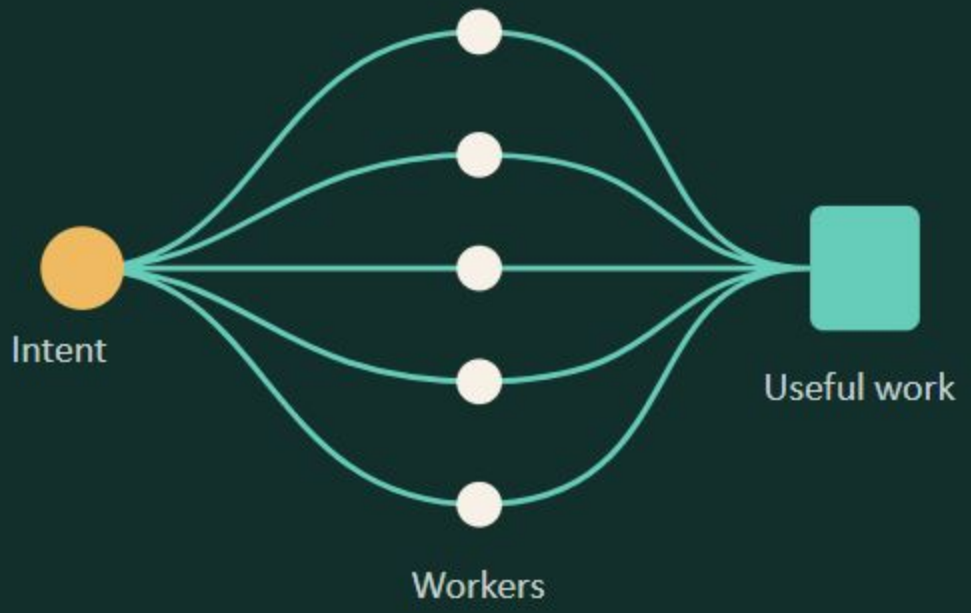
Braxton • Home lab systems & AI • September 17, 2026

One human. Many agents.

Keep your project coherent.



How do you keep it coherent—and still enjoy operating it?

The project changes. Do the rules?

01

A script

Becomes a service.

Who depends on it now?

02

A weekend automation

Becomes a household dependency.

What happens when it stops?

03

A learning exercise

Becomes a long-running project.

What changed about the job?

Which assumptions from day one are still running your project?

Where will another agent help?

Separate the questions.

Independent investigations
Distinct artifacts
Different evidence paths

Overlap the waits.

A bounded job runs.
Another useful task
can move forward.

A → B → C

Keep dependencies clear.

Shared state and ordered
changes need coordination
before more workers.

Can you explain the boundary—and still review what comes back?

A correction has to change the next action.

Where does the fix live?

In the record—and in the next action?

We notice a mistake

Record it

Keep the evidence.

→

We decide what to change

Repair the mechanism

Owner, check, tool, or workflow.

→

We encounter it again

Observe the difference

Did the next action change?

A record preserves the lesson. A working mechanism carries it forward.

Plan for the next handoff

One month

Finish and hand over.

- Bound the outcome.
- Make checks and recovery repeatable.

Can another actor finish this?

Six months

Continue through change.

- Preserve decisions and dependencies.
- Budget review and maintenance.

Can another session continue?

One year

Survive replacement.

- Own services and migrations.
- Test restore and exit paths.

Can the project outlive its setup?

Plan near-term work in detail; name distant assumptions and revisit triggers.

Who is doing the coordinating?

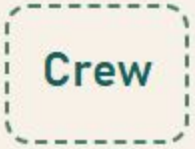
Human-directed lanes

Assign work.

Read the evidence.

Integrate the result.

Who integrates the results?



A persistent crew

Shared tasks.

Durable memory.

Continuity across sessions.

What survives the session?

A supervised loop

Observe runtime state.

Act within a grant.

Check what changed.

Who notices and acts?

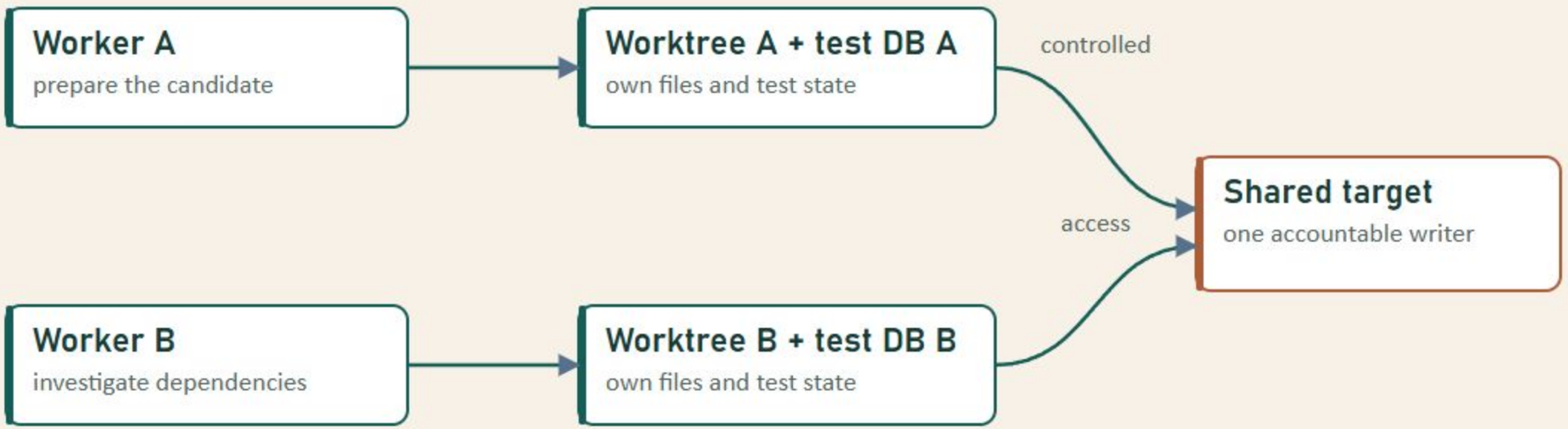
Coordination work moves. It never disappears.

Give the worker a contract

Service upgrade	Preparation worker
Outcome	A reviewable upgrade candidate; consumers still work.
Inputs	Pinned revision, dependency list, acceptance checks.
Territory	Own worktree and test environment.
Authority	Prepare and test. Production execution is separate.
Stop / budget	Stop on unknown data migration or missing recovery.
Done when	Candidate + evidence + limitations + next action.

An assignment is a small operating agreement.

Own every mutable boundary



Files, databases, ports, credentials, and targets each need an owner.

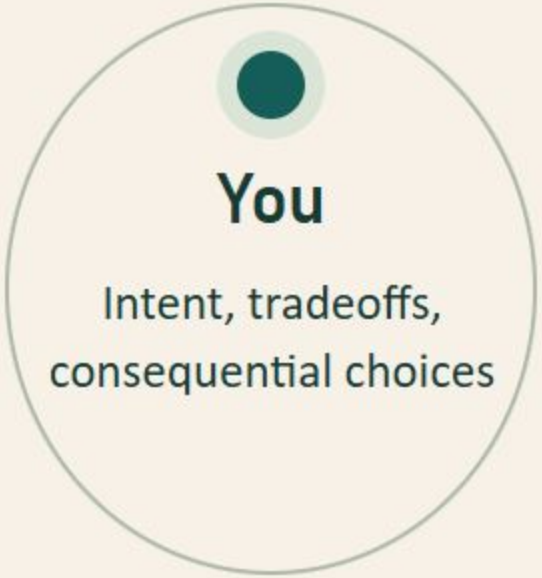
Human attention belongs in the architecture.

Routine work

Bounded grants

Target • action • limits

Revocation • evidence



Keep challenge useful.

The agent can be wrong.

The human can be wrong.

Either can get stuck.

What reaches you

that should not?

What should reach you

but does not?

Keep the history. Design the working view.

Experiments

Errors + corrections

Decisions + reasons

Receipts + provenance

Durable history

Ask

What do I need to do this job?

Identity • authority

freshness • provenance

Current task

What is true now?

What can I change?

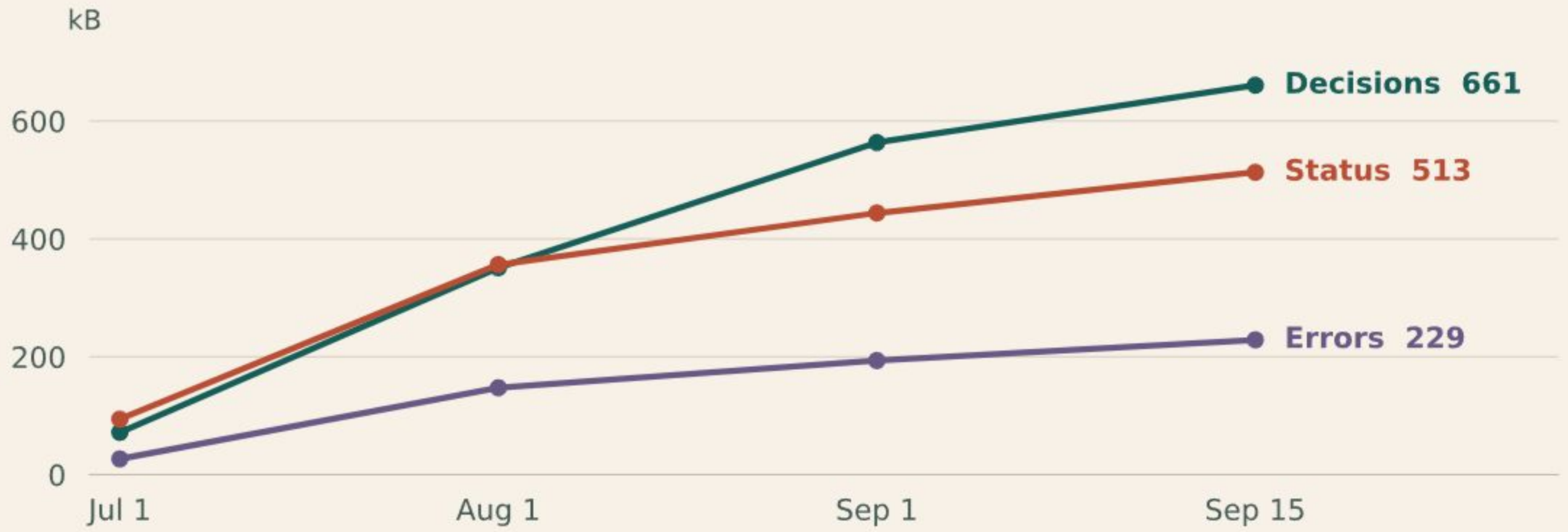
What happens next?

Link back to the evidence.

Can you find what to do now without rereading everything that happened?

Concept view • Advance for a measured example

Keep the history. Design the working view.



One project, three files • decimal kB • snapshots before the labeled UTC dates

Measure your required reading—and the time to find the next action.

Example view • Advance to the cold-pickup test

Make the handoff executable

The worker disappears.

Can the next actor continue?

Leave a usable state packet

- Current task, owner, and revision
- Evidence and last observed target state
- Pending decisions and granted authority
- Next action and recovery instructions
- Tools and access needed to perform them

Test a cold pickup with only the durable record.

Supervise the work that is running

Present	Process or session exists
Bound	Correct task, revision, environment
Progressing	New evidence or a clear blocker
Complete	Acceptance evidence is available

Observe → decide → act → observe

A reassignment is incomplete until the worker follows it.

A fresh state pass can give the human a useful re-entry point.

The supervisor's own actions need feedback.

Could this check ever have failed?

Linux AER counters

reported errors

Direct PCIe status:
replay timeout

A quiet counter was not a clean path.

Known good → pass Known bad → fail

Revisit the raw receipt.

Repair the check.

Test the check against the failure it claims to exclude.

Complete the change

Illustrative replay • synthetic records • no live target

Hold | Correct | Recheck | Apply | Accept

Hold

Dependency check fails

The candidate omits a utility required by another service.

Preserve the finding. Do not execute.

Advance to walk through the correction and acceptance.

Complete the change

Illustrative replay • synthetic records • no live target

Hold | **Correct** | Recheck | Apply | Accept

Correct

A new candidate is prepared

Repair the dependency plan; attach the revised artifact and recovery steps.

Earlier evidence stays attached to its earlier candidate.

Advance to walk through the correction and acceptance.

Complete the change

Illustrative replay • synthetic records • no live target

Hold | Correct | **Recheck** | Apply | Accept

Recheck

Fresh evidence passes

Repeat the consumer check and the deliberately broken control in staging.

Review the exact candidate before execution.

Advance to walk through the correction and acceptance.

Complete the change

Illustrative replay • synthetic records • no live target

Hold | Correct | Recheck | **Apply** | Accept

Apply

Execute the bounded grant

Record authorized intent and the attempt first. Apply the candidate, then observe and record the outcome.

If reporting is interrupted, observe the target before retrying.

Advance to walk through the correction and acceptance.

Complete the change

Illustrative replay • synthetic records • no live target

Hold | Correct | Recheck | Apply | **Accept**

Accept

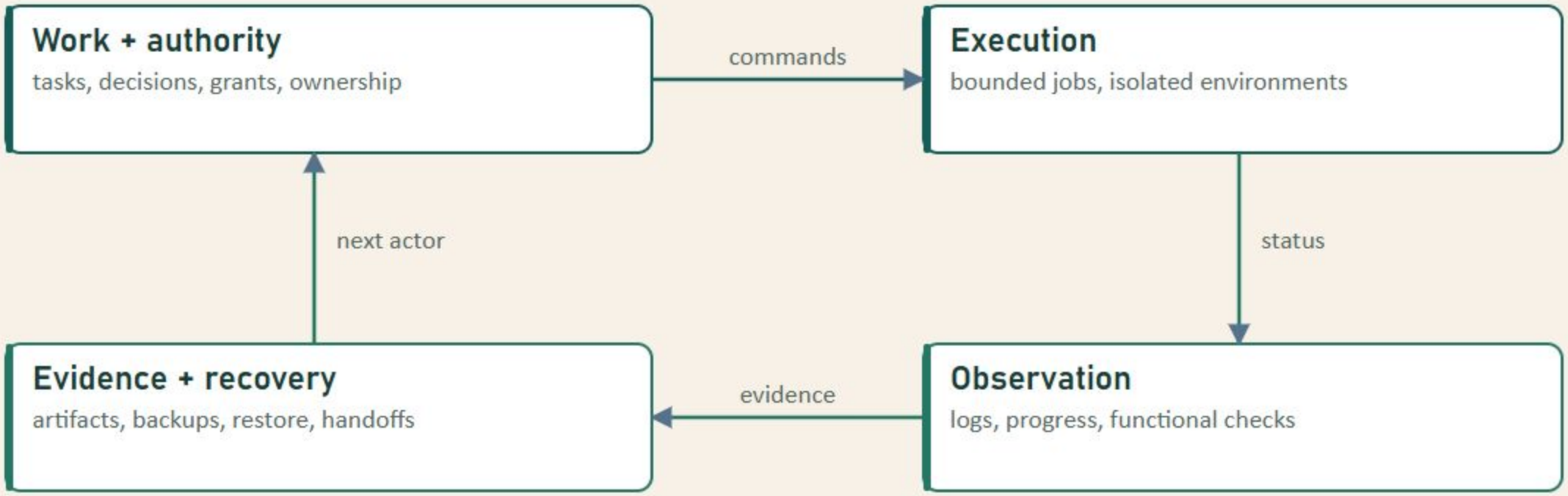
Consumers work; the record agrees

The functional check passes, the deployed state is recorded, and acceptance is explicit.

A useful result, ready for the next actor.

Advance to walk through the correction and acceptance.

Give each service a job



These are logical responsibilities, not a required machine per box.

Make tomorrow-you a good teammate.

01 / ARRIVE

Get your bearings.

Project status

Task + revision
What changed
Running jobs
Needs a decision

One command or current page.
Show when it was checked.

02 / WORK

Give inputs a home.

One inbox

New material → next review
Questions → one digest

Temporary context
Hide after Friday

Name what interrupts now.
Let temporary notes expire.

03 / LEAVE

Leave a way back.

Next session

Changed: candidate + evidence
Open: restore check
Next: test recovery
Still running: none

A short note to your next self.
Keep the evidence linked.

Pick one friction point. Make the useful habit easier to do.

What should your project keep or change?

Keep

What earned its place

Evidence that can be revisited

Checks that expose mistakes

Clear ownership and recovery

Change

What gets in the way

Rules that never reach the workflow

Views nobody can keep current

Dispatch beyond review capacity

After the change, ask:

Can I operate it?
Can we finish useful work?

A project you can operate—and enjoy.

More room to explore. A way back to the work.

Take one piece home.

One habit: leave a way back.

One guardrail: bound the change.

One test: can a fresh actor continue?

Try it on a task you already care about.

A lease is only part of ownership

Optional depth

Claim

Check run state

Recognize duplicate requests

Reject a live competing lease

Record claimant and expiry

After expiry

The old worker may still run.

The destination must reject stale authority or serialize consequential writes.

A claim record, a queue entry, and an external effect solve different problems.

Give state one authoritative home

Optional depth

Narrative

Why and how

Decisions, explanations, handoffs, runbooks.

Transactional state

Who owns what now

Claims, attempts, transitions, identities.

Derived views

How we find and observe

Search, dashboards, summaries, telemetry.

Preserve identity and provenance when information crosses stores.

The same shape, different acceptance

Optional depth

Read-only audit

Pinned inventory

Re-runnable checks

Findings with limits

Experiment

Registered question

Inputs + environment

Result + interpretation

Service deployment

Authorized target

Functional checks

Observed state + recovery

Choose the evidence that establishes the intended outcome.

Choose tools by their responsibility

Optional depth

Git + forge + CI	Version, review, repeat checks
SQLite / PostgreSQL	Own structured state and transactions
Process manager / workflow engine	Execute; add durable recovery when needed
Files / object storage	Retain artifacts and restore evidence
Logs / OpenTelemetry pipeline	Observe work across services

A new component adds an operating obligation as well as a capability.

References and questions to take home

Optional depth

Official technical references

Git: linked worktrees

PostgreSQL: queue-oriented locking

Temporal: activity idempotency

OpenTelemetry: collection and export

Test your own project

Can a fresh actor take over?

Can a check catch a known defect?

Can the target reject stale authority?

Can recovery work without the agent?