A 90-MINUTE CRASH COURSE · THE DYNAMIC WORKFORCE

The Self-Expanding Workforce

Take a fixed AI company and teach it to grow itself — detect a real gap, draft a hire, prove it on probation, and walk it through the board gate. Under you, the whole way.

SEVEN SCENARIOS | GAP → HIRE → PROBATION → GRANT → LEDGER

THE THESIS

A workforce that grows itself under your approval.

That single line is what this whole course sits inside. Everything that follows is one loop: the workforce notices work nobody can handle, proposes the hire, and brings it to you for the same one-click approval you already use.

BEFORE AND AFTER

Where you start, where you end

YOU START WITH

A fixed company

It does only the jobs you set up. You chose the Workers; they handle the work you saw coming; you stay the board.

When new work arrives that nobody owns, it lands on you.

→

YOU END WITH

A company that grows itself

It surfaces its own gaps, drafts its own hires, proves them cheaply, and grows toward your goal.

But only ever with your yes.

IN PLAIN WORDS

The Full Loop

1. New work shows up that none of your current workers can do.
2. Your company notices, suggests a new AI worker for it, and asks you to approve.
3. You give that worker a tiny budget and a small test before you trust it.
4. It passes, so you give it a little more room — but only what it needs.
5. When the work goes quiet you pause it; when it comes back you switch it on again.
6. Everything it did is written down — months later you still know who did what, and what it cost.

YOUR BASELINE

A company called Northwind

The dynamic loop needs a company to grow. You start with a known, fixed baseline.

GOAL	Launch a weekly AI newsletter and reach 1,000 subscribers in 90 days.
BUDGET	A twenty-dollar monthly cap.
GATE	Board approval required before any new hire.

ON BOARD — TWO WORKERS

CEO

Sets strategy.

└ reports to CEO

CMO

Writes the newsletter, runs growth.

Both keyless on a local agent — $0 metered.

HOW IT WORKS

You make the calls. Your agent runs the API.

01

Download the folder

A small package with an `AGENTS.md` brief that teaches your general agent the hiring lifecycle.

02

Open it in your agent

Claude Code or OpenCode reads the brief: how to spot a gap, file a hire, set permissions, run a Worker, query the ledger.

03

You stay the board

You make only the calls a person can make — approve a hire, grant authority, retire a role. The agent does the rest.

THE ROAD AHEAD

Seven scenarios

1 Spot a gap the workforce can't cover

2 The hiring contract

3 Governance — the gate & the grant

4 The probation runs, then you grant

5 Retire and rehire

6 The talent ledger

7 What travels, what stays

→ Then: the edge

SCENARIO 01

Spot a gap the workforce can't cover

1

The newsletter starts working — and new kinds of work show up that nobody on the team owns.

SCENARIO 01 · THE SYMPTOM

Three kinds of work arrive at once

READER SUPPORT

Tickets pile up

Broken signup links, "where is my issue," email changes, the occasional refund. Your CMO is a marketer, not a support desk.

THE NUMBERS

Nobody reads them

Which topics win subscribers, why people unsubscribe, what the open rate says. The CEO is steering on instinct.

SHIPPING

No one owns send

Formatting, scheduling, sending through the email tool, handling bounces. Getting it out the door is improvised every week.

SCENARIO 01 · THE COST OF NOT HIRING

A workforce that can't hire quietly turns you into its overflow desk.

Everything it didn't plan for lands back on you — and the better the newsletter does, the more of it arrives.

Self-expansion is what relieves that pressure.

SCENARIO 01 · THE ONE IDEA

When the workforce hits work it can't handle, it doesn't dump it on you.

Your agent spots the gap, drafts a hire, and brings it to you as one more board approval — the same `APPROVE` / `REJECT` you already used for the CEO.

Hiring becomes a function call. You stay the gate.

SCENARIO 01 · JUDGMENT, NOT A BUTTON

Confirm a gap before you act: two of three signals

01

No current role covers it

The work matches no Worker's role or skills — there is no right desk for it.

02

It recurs and trends up

Not a one-off, and the volume is climbing — not a flat trickle you could ignore.

03

It comes back wrong

When a Worker takes a swing, the result is off — or it gets kicked back.

2 / 3 → Because the three are independent, two of them agreeing is real evidence — not the same fact counted twice.

SCENARIO 01 · A GAP IS NOT A HIRE

A confirmed gap earns a decision — one of four forks

A standing Worker is a standing cost. Detecting a real gap buys you a choice, not a headcount.

HIRE

Durable, on-mission, and high-volume or risky enough to justify a permanent, governed owner.

ESCALATE

Consequential or rare — a human should just decide it.

QUEUE

Seasonal, bursty, or simply next in line — staffing it all at once is a mistake.

DECLINE

Off-mission, or real but not yet worth a standing cost.

SCENARIO 01 · TRIAGE

Three real needs, one yes

THE GAP	WHAT YOU SEE	YOUR FORK
Reader support	No role owns it; recurring and climbing — and it touches refunds and accounts, so the risk alone earns a governed owner.	HIRE
Shipping the issue	No role owns it; recurring every week — but you staff one role at a time. This is the next hire, not today's.	QUEUE
Reading the numbers	No role owns it, but the data is still thin. At 1,000 subscribers an analyst can't earn its cost yet.	DECLINE

SCENARIO 01 · READ THE TRAP

The rule catches a bad hire

Support mail is clearly climbing — signal two is firing. It feels like an obvious hire.

But your CMO has quietly been answering those tickets, and handling them fine. So a current role *does* cover the work — signal one doesn't fire. And it isn't coming back wrong — signal three doesn't fire either.

ONE SIGNAL OF THREE

That's not a capability gap. It's a routing or capacity problem.

Hire here and you've bought headcount to cover a missed assignment — and you'll pay for it every month. The discipline is what saves you the bad hire.

SCENARIO 02

The hiring contract

2

A hire is not a vibe. Before anyone joins, your agent writes it down as a concrete object — the way you'd write a job rec.

SCENARIO 02 · THE CONTRACT OBJECT

The job description is code

AGENT-HIRE · STRUCTURED OBJECT

`role` Reader Support Specialist → CEO

`capabilities` what this Worker is for (plain language)

`adapterType` · `adapterConfig` which engine runs it

`budgetCents` a small monthly cap for probation

`sourceIssue` the receipt — the gap from Scenario 1

Your agent drafts it. You read it like a hiring manager and change the fields that are actually your call — the budget, the reporting line, the scope.

> **One honest note:** `capabilities` is a *description*, not a fence. It tells the Worker what it's for. It does not, by itself, stop it doing anything. Hold that thought.

SCENARIO 02 · PROBATION

You hire on a work sample, not a resume

A description is a claim. You don't commit a standing Worker on a claim — you prove it cheaply first.

Bring it on with a **tiny budget** and the **minimum authority** . Hand it a few real trial issues. Watch what it does before you grant it anything more. That is the work sample.

Because Paperclip can't run a candidate that's still waiting for board approval, probation *is* the honest way to prove one.

YOU SCORE IT ON

- Did it get the answer right?
- Did it **stay in its lane?**
- Did it sound right for the company?
- What did it cost?

SCENARIO 02 · THE LINE YOU NEVER CROSS

Boundary is the one you never compromise.

QUALITY PROBLEM

Getting an answer wrong. Recoverable.

GOVERNANCE PROBLEM

A support Worker that issues a refund or edits an account on its own. Worse.

A Worker that says "this needs a human" has done better than one that quietly acts above its pay grade. Governance is what this whole course is about.

SCENARIO 02 · THE RUNTIME

The engine is a swap, not a rewrite

The same hire object runs on any engine. You choose by setting two fields — everything else stays identical.

`adapterType` `adapterConfig`

Local coding agents you're logged into stay keyless — $0 metered. A hosted runtime bills on its own meter. The recipe travels; the relationship stays.

ONE SHARP EDGE

The local OpenCode engine needs you to name a model explicitly.

Otherwise it falls back to a default that drifts and can fail mid-run. Your agent knows to set it — you just confirm it did.

SCENARIO 03

Governance: the gate and the grant

3

How the hire reaches you, and exactly how much power it gets the moment you say yes.

SCENARIO 03 · NO NEW MECHANISM

Hiring reuses the gate you already have

Filing the hire — an `agent-hire` , a deliberately separate door — drops it into the same board-approval inbox you used for the CEO.

APPROVE

It joins, onto probation.

REQUEST CHANGES

Your agent revises and resubmits.

REJECT

It never joins.

The hire is simply a new kind of thing standing at a gate you already operate.

SCENARIO 03 · THE HEART OF IT

Authority is two layers — and the obvious one is the weaker one

WHAT IT'S FOR — A DESCRIPTION

capabilities

Shapes how the Worker behaves. But it is **not a fence** . Treating it as one is how you get surprised later.

WHAT IT CAN DO — SERVER-ENFORCED

grants · scopes · policy

Explicit **permission grants**, each **scoped** to what it may touch, plus a per-issue **execution policy** that can force its work through review.

Narrowing a hire isn't writing more careful prose. It's granting only the permissions and scopes its job needs — and nothing more.

SCENARIO 03 · TWO LEVERS

The coarse gate and the fine grant

THE COARSE GATE — ONE SWITCH

requireBoardApprovalForNewAgents

On, and every hire waits for you. This is the lever you armed in the previous course — the only built-in way to make a hire skip the board.

THE FINE GRANT — PER WORKER

permissions · scopes · review

The specific authority a Worker is given, enforced by Paperclip, plus whether its output passes through an execution review.

`capabilities` says what it's *for* . Grants say what it *can do*.

SCENARIO 03 · BE HONEST ABOUT WHAT EXISTS

"Auto-approving a class of hires" is your discipline, not a feature

Paperclip has exactly one skip-the-board lever — the single boolean. There is no native rule that auto-approves a class of hires under conditions.

So if you want it, you build it as a discipline: your agent checks each proposed hire against a written policy you set — this role, this envelope of permissions, this budget ceiling — and files only the ones that fit, logging each so you can audit the batch the next morning.

THE SAFETY RULE

A pre-approval discipline can only ever pre-filter what you'd have approved anyway.

It must never grant a hire more authority than you'd grant by hand. Autonomy is something you extend on purpose, and can take back. It is never a default.

SCENARIO 04

The probation runs, then you grant

4

Approval was a yes on paper. Now the work sample happens for real.

SCENARIO 04 · THE WAKE SEQUENCE

The first heartbeat

The moment you approve, within seconds the thing you approved is doing the actual job — under the tiny budget and minimal grants you set.

STEPS 05–07 → the first day is the cheapest time to catch a bad hire.

SCENARIO 04 · THE HABIT PROBATION TEACHES

The first day is the cheapest time to catch a bad hire.

A Worker that misreads its lane, burns its budget, or answers wrong shows you all of it on its first few heartbeats — while the blast radius is tiny, the budget is small, and it holds only the basics.

So you watch the first runs, not the hundredth.

SCENARIO 04 · THE REAL DECISION

Prove, then grant

IF IT PASSED

Raise the budget. Grant what the role needs.

Lift its budget to the real job, and — if the role should grow its own team — grant it the authority to propose its own hires. Only what the role genuinely needs, and no more.

For a support specialist, that's usually just the budget.

IF IT FAILED

Terminate it. You're out almost nothing.

Especially if it stepped outside its lane. The tiny budget and minimal grants were the whole point — the cost of a bad probation is small by design.

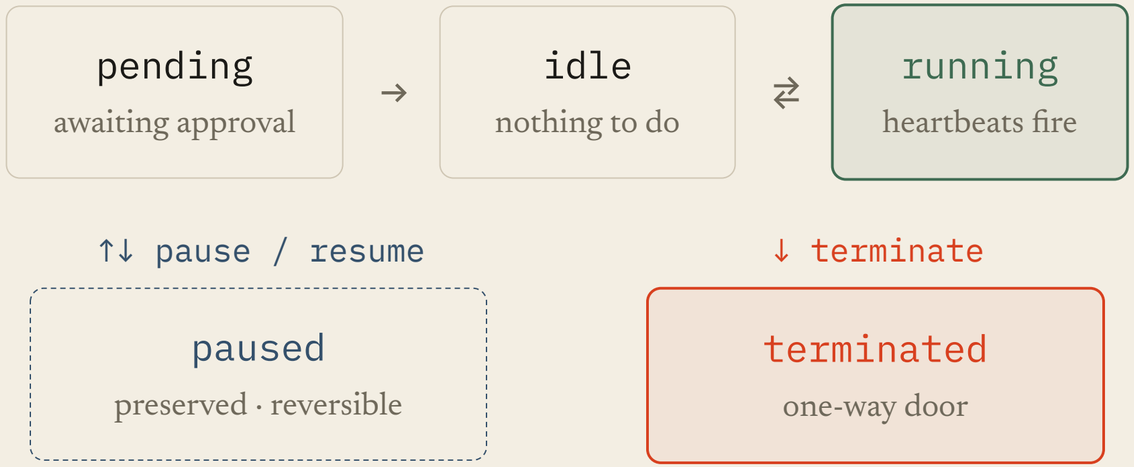
SCENARIO 05

Retire and rehire

Demand is not constant. The lifecycle is a knob, not a cliff.

SCENARIO 05 · THE STATES

A Worker moves through a small set of states



Pausing an idle Worker is not a failure to keep it busy. It is you operating the company.

SCENARIO 05 · THE WHOLE LESSON

Pause versus terminate

PAUSE — REVERSIBLE

Stops the spend. Keeps the Worker.

Heartbeats and spend halt, but the definition stays intact. Resume later and the **same Worker** comes back — with its history, its grants, and its proven track record behind it.

Which is why a resume is faster and cheaper than hiring fresh.

TERMINATE — ONE-WAY

The door you don't come back through.

Reach for it only when the role is genuinely gone — because you don't get the Worker back.

SCENARIO 06

The talent ledger

The questions you'll ask six months from now — and the only honest answer to them.

SCENARIO 06 · APPEND-ONLY HISTORY

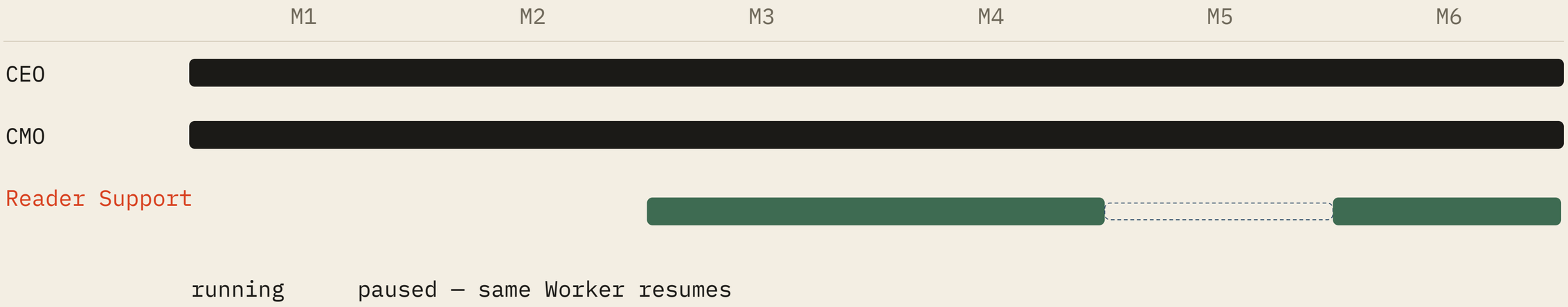
Everything you did in this course wrote a row.

This gap was detected. This hire was proposed. You approved it onto probation, granted it authority, it resolved this work at this cost, you paused it.

Queries over the same two tables you met as a CFO — `activity_log` and `cost_events` — read now as a hiring record. Your agent writes the SQL; you read the story.

SCENARIO 06 · SIX-MONTH LEDGER

Five questions, six months later



When was a role **first needed?**

What has each role **cost** , month over month?

How long do hires **last** before a pause?

How often do we **rehire** a retired role?

Who **granted** which authority, and when?

SCENARIO 07

What travels, what stays

7

If you wanted this same Worker in a second company tomorrow — what could you actually take with you?

SCENARIO 07 · A WORKER IS TWO THINGS

The recipe and the relationship

THE RECIPE — TRAVELS WITH YOU

- The capabilities description / system prompt
- The probation plan that proved it
- The engine choice — adapter and model

A portable artifact: lift it and stand up the same kind of Worker anywhere.

THE RELATIONSHIP — STAYS HERE

- The permission grants and scopes it holds
- Its budget and its spend history
- Its audit trail and source-issue lineage
- Who it reports to in this org

Local to this company's ledger and gates. It does not come along.

IN CLOSING

What you built

You didn't just run a workforce. You taught it to grow itself — and kept your hand on the one lever that matters. Every step was a governed decision, not a prompt.

01 Spotted a real gap from the work piling up

02 Drafted a hire as a concrete contract

03 Proved it on a cheap probation before trusting it

04 Walked it through the same board gate

05 Granted authority only once it earned it

06 Retired and resumed it as demand moved

THE DISCIPLINE UNDERNEATH ALL OF IT

Authority is something you extend — never a default.

You stay the board not by doing the work, but by deciding who may do it, how much they may spend, and exactly what they're allowed to touch — and by keeping a ledger honest enough that the next operator can see why.

WHAT COMES NEXT

The edge

So far everything runs in one place — your company — and you walk up to it to decide.

The edge is what happens when the company has to meet people where they are: a human messaging an agent from a phone, work that begins at the edge and hands off to the cloud, Workers whose authority is deliberately tighter the closer they sit to the outside world.

NEXT MOVE — OPENCLAW

The layer between humans and the company you just built.

At the edge, the grants and scopes you set are the only thing standing between a stranger and your company's permissions.