

The AI-ready team playbook

Six layers your product and engineering
teams write into the repo, and own.



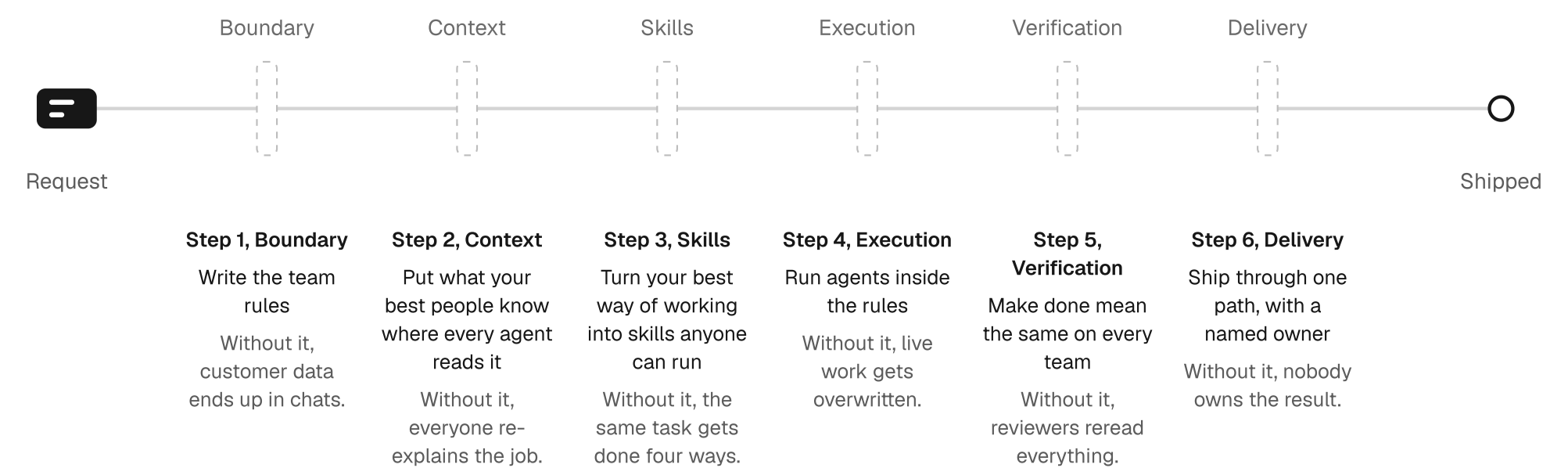
Contents

Six layers your product and engineering teams write into the repo, and own.

Before the gates	
Six gates from request to release	3
Why teams stall at Assisted	4
Find your stage	7
The six gates	
 Step 1 Write the team rules	8
 Step 2 Put what your best people know where every agent reads it	10
 Step 3 Turn your best way of working into skills anyone can run	12
 Step 4 Run agents inside the rules	14
 Step 5 Make done mean the same on every team	16
 Step 6 Ship through one path, with a named owner	18
After release	
Make it stick: train the people who keep it running	20
Questions leads ask	22
Terms	23
Train your team on your own repo	24

Six gates stand between a request and a release

Follow one change through this playbook: Acme Inc's checkout squad asks for annual billing. It has to pass six gates on its way to release, and each step chapter frees it at one.



Before the gates

Why teams stall at Assisted Find your stage

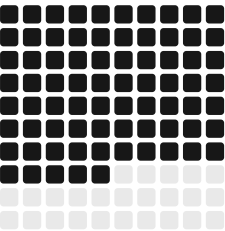
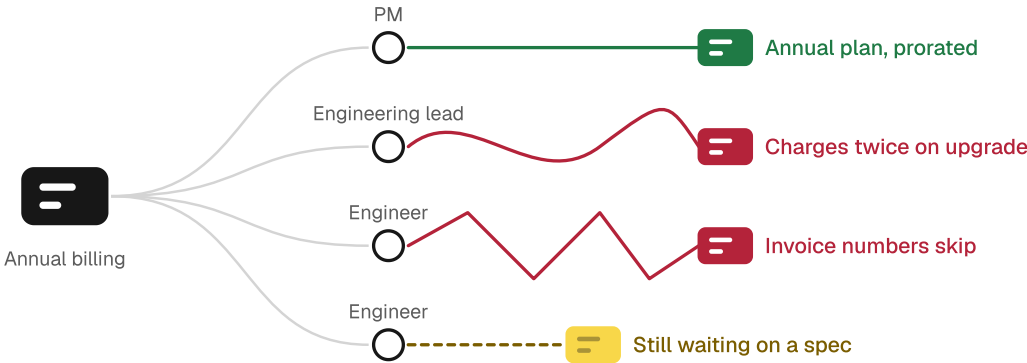
After release

Make it stick: train the people who keep it running Questions leads ask
Terms Train your team on your own repo

Everyone uses AI. Almost nobody wrote down how.

Most product and engineering teams already use AI every day. Few have written down how, so each person works their own way and delivery does not speed up.

The squad gets one request: launch annual billing. Four people each ask their own AI in their own way, and four different results come back. Nothing one of them learns reaches the rest.

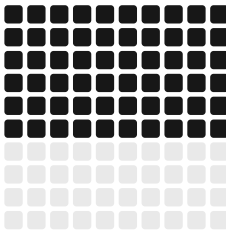


75%

of knowledge workers use AI at work.

Usage is already there.

Source: Microsoft and LinkedIn, 2024 Work Trend Index



60%

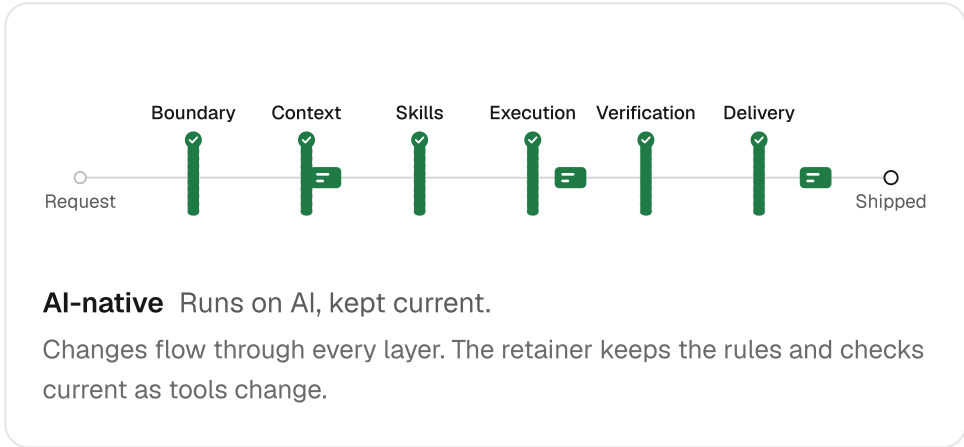
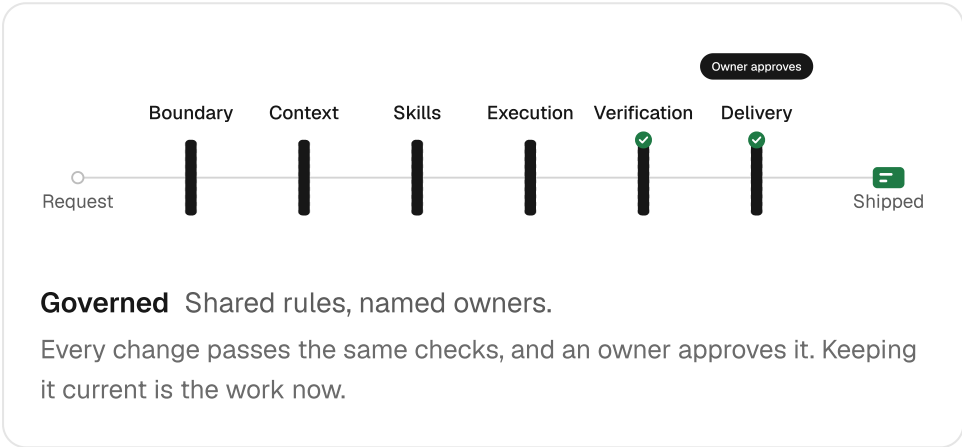
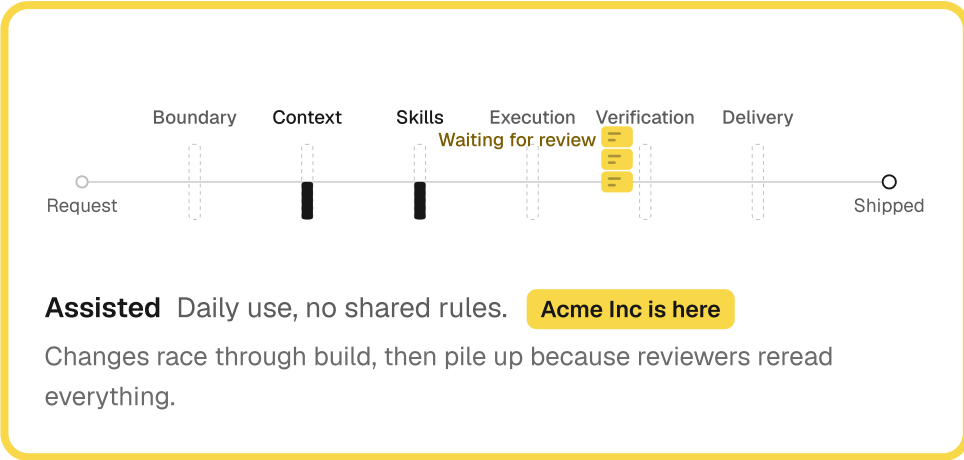
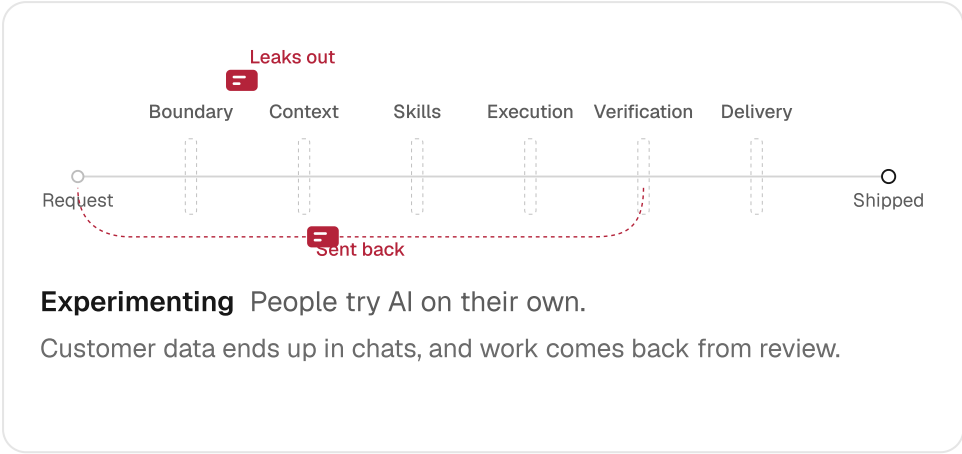
of leaders worry their organization's leadership lacks a plan and vision to implement AI.

The written plan is what is missing.

Source: Microsoft and LinkedIn, 2024 Work Trend Index

Teams stall at the second stage, where AI is daily but private

The same change on the same track, at each of the four stages. Acme sits at Assisted: everyone uses AI every day, and the changes pile up in review because nothing is shared.



The way out is six layers your team writes into the repo

Each layer is a few plain text files. Anyone can read them, copy them and improve them, the way open source works. The six step chapters write them one gate at a time.



Mark where each team is before anyone trains

Before you train anyone, write down where each team is today. A baseline turns "we use AI" into a stage you can move.

Who	Heads of product and engineering, with each team lead
How long	Twenty minutes per team

Pause point

Nobody trains until every team has a stage marked.

Keep it current

- ☐ **Before you start** Both heads agree on one goal before anyone writes a rule.
- ☐ **Six weeks later** Mark the sheet again, so the change shows.



Team baseline

Team

- 1

Is there one written rule, approved by a person, for what agents may see and change?

YesNo
- 2

Does each task pull its context from one shared source that someone keeps current?

YesNo
- 3

Are your best people's ways of working written down as skills anyone can run?

YesNo
- 4

Does every piece of AI work pass agreed checks before a person reviews it?

YesNo
- 5

Does every change ship through one path, with a named owner approving it?

YesNo

Mark the team's stage

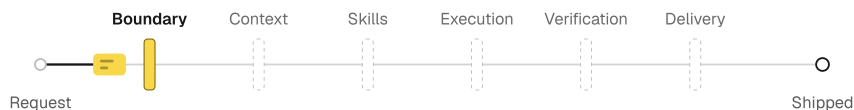
☐ Experimenting

☐ Assisted

☐ Governed

☐ AI-native

Mostly no means Experimenting or Assisted. Mostly yes means Governed. The assessment scores all six layers in two minutes. Take the assessment



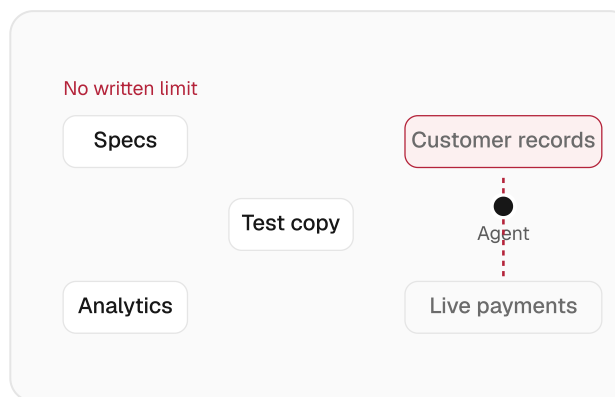
Step 1 of 6 Write the team rules

Agents go wherever nobody wrote a limit

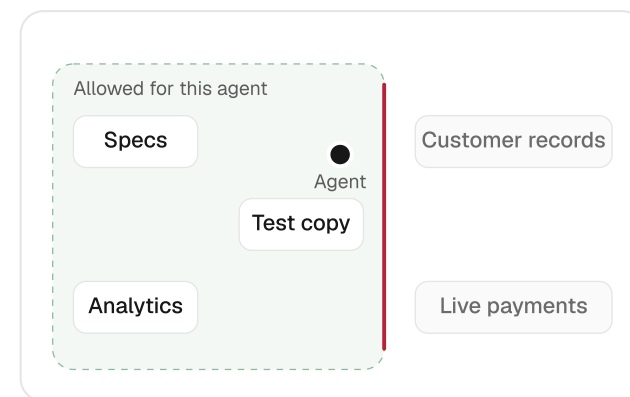
The change stops at the first gate: nobody wrote down what agents may read, so customer records are as open as anything else. Boundary is what each agent may see and change, written down and approved by a person.

What to do

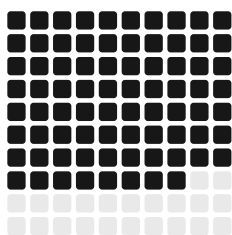
- 1 List the data each team works with, and mark what no agent may read.
- 2 Write what agents may change, and where they must stop and ask.
- 3 Have the heads of product and engineering approve it, by name.



Today: no written limit, so the agent reads customer records like any other file.



With the rules: the agent works inside what was approved and stops at the edge.

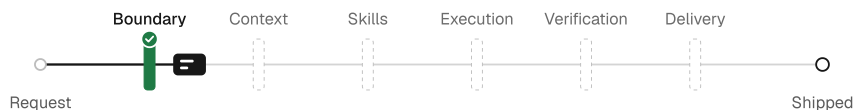


78%

of AI users bring their own AI tools to work.

Personal tools on company data is the risk the rules close.

Source: Microsoft and LinkedIn, 2024 Work Trend Index



Write it down: .agents/settings.json

Acme Inc repo

acme-checkout/

.agents/

agent-memory/ 1 file

agents/ 1 file

rules/

o billing-service.md

● customer-data.md

o testing.md

● settings.json

skills/ 5 files

o AGENTS.md

db/

docs/

exports/

services/

web/

.agents/settings.json

```

1 {
2   "permissions": {
3     "allow": ["Bash(npm run test:*)"],
4     "ask": ["Edit(db/migrations/**)",
5     "deny": [
6       "Read(/.env*)",
7       "Read(/exports/customers/**)"
8     ]
9   }
10  }

```

.agents/rules/customer-data.md

```

1 ---
2 paths: ["services/billing/**", "db/**"]
3 ---
4 Agents read the seed copy in db/seed/ only.
5 Names, emails and cards never enter a prompt.
6 Approved: head of engineering, with legal.

```

Owner: head of engineering, with legal

Nobody has to guess what is safe to paste.

Who

Heads of product and engineering

How long

About an hour

Pause point

Nobody pastes customer data into an agent until this file is approved.

Keep it current

☐ Each quarter Read the rules again and sign them again.
 ☐ A new data source arrives Add it to the file before any agent sees it.

Acme Inc after step 1

Example

Boundary

Context

Skills

Execution

Verification

Delivery

● ● ● ○ Governed

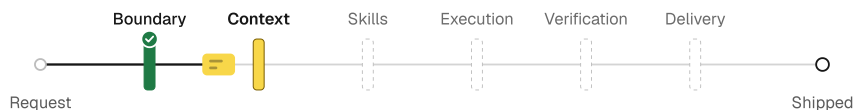
● ● ○ ○ Assisted

● ● ● ○ Governed

● ● ● ○ Governed

● ○ ○ ○ Experimenting

● ● ○ ○ Assisted



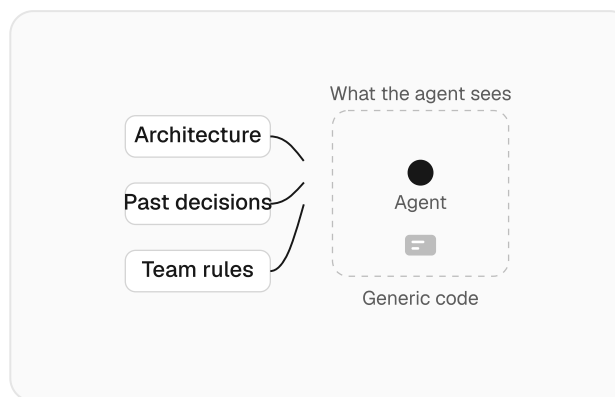
Step 2 of 6 Put what your best people know where every agent reads it

Agents start from zero when knowledge lives in heads

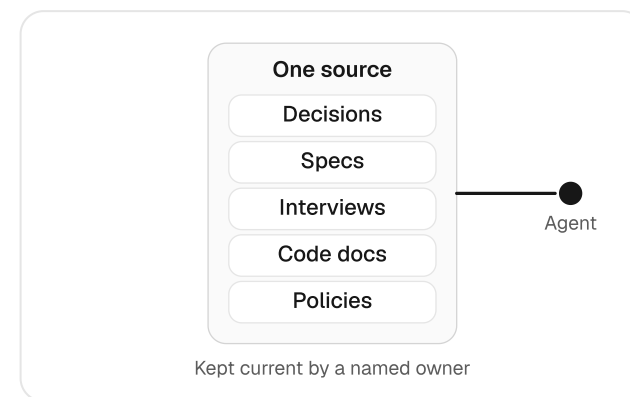
Past the first gate, the change stalls again: the agent never saw the decisions behind annual billing, so its code is generic. Context is the knowledge each task needs, kept in one place someone maintains.

What to do

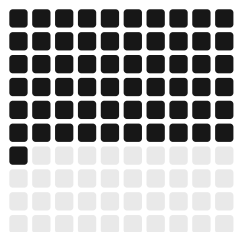
- 1 Pick the five documents every spec and change needs, and move them into the repo.
- 2 Give each source a named owner who keeps it current.
- 3 Keep AGENTS.md short: cut any line whose removal would not cause a mistake.



Today: architecture, past decisions and team rules sit outside what the agent sees, so the code comes out generic.



With one source: scattered notes gathered into one place a named owner keeps current, and every agent reads it.

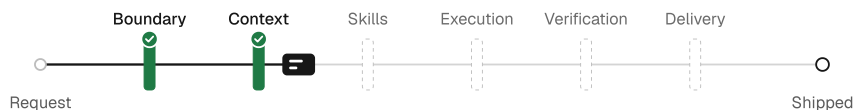


61%

of developers spend more than 30 minutes a day searching for answers or solutions to problems.

Shared context gives that time back, to people and to agents.

Source: Stack Overflow, 2024 Developer Survey



Write it down: AGENTS.md

● ● ● Acme Inc repo

acme-checkout/

.agents/

agent-memory/1 file

agents/1 file

rules/

● billing-service.md

● customer-data.md

○ testing.md

● settings.json

skills/5 files

● AGENTS.md

db/

docs/

exports/

services/

web/

AGENTS.md

1 # Acme checkout

2 Annual billing ships before usage pricing.

3 ## Commands

4 npm run test:billing (40 seconds)

5 npm run lint

6 ## Style

7 Money is integer cents, never floats.

8 Prices come from plans.ts, never typed.

9 ## Gotchas

10 Invoice numbers never skip or repeat.

11 Read docs/decisions/ before a billing spec.

.agents/rules/billing-service.md

1 ---

2 paths: ["services/billing/**"]

3 ---

4 Call billing through its client, never its tables.

5 Proration rounds down to the cent.

Owner: engineering lead, with the product lead Every agent starts from what is already decided.

Who Engineering lead, with the product lead

How long Half a day for the first five sources

Pause point An agent task starts only from sources with a named owner.

Keep it current

- ☐ **A decision changes** Update the source the same day, in the same change.
- ☐ **Each month** Delete what nobody read. Old context misleads.

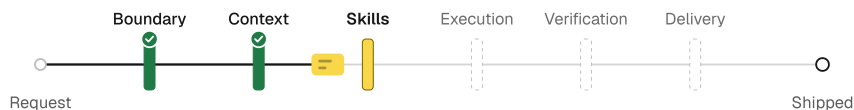
Acme Inc after step 2

Example

Boundary	● ● ● ○	Governed
Context	● ● ● ●	Governed
Skills	● ● ● ○	Governed
Execution	● ● ● ○	Governed
Verification	● ○ ○ ○	Experimenting
Delivery	● ● ○ ○	Assisted

KaaS.team

11 of 24



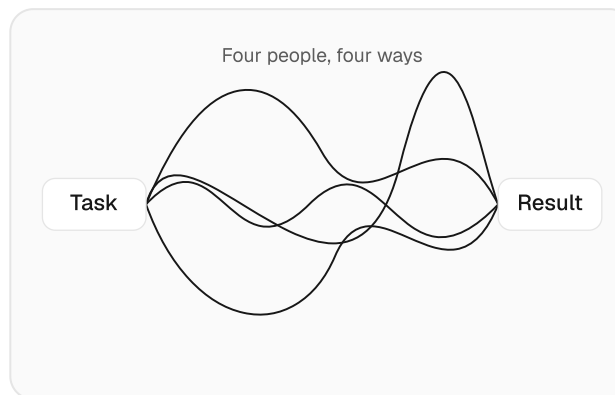
Step 3 of 6 Turn your best way of working into skills anyone can run

Four people do the same task four ways

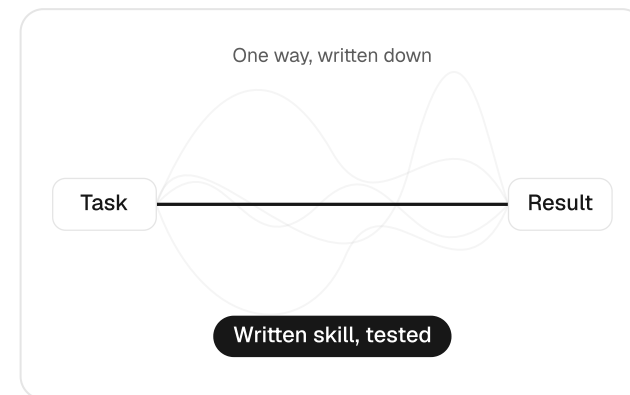
Four people write the launch brief four ways, and each one comes out different. A skill is how your best people do a task, written down so every person and every agent does it that way.

What to do

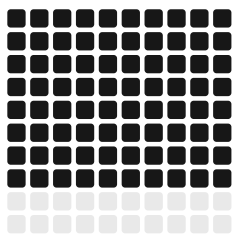
- 1 Pick one task your teams do every week, such as a launch brief or a billing review.
- 2 Ask the person who does it best to write the steps, and what good looks like.
- 3 Run it on a real task, then fix the skill from what the review found.



Today: four people, four routes, four different results for the same task.



With a written skill: one tested way, so the result is the same whoever runs it.

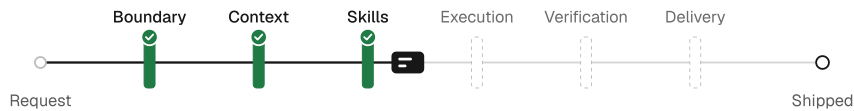


4 in 5

people say they want to learn more about how to use AI in their profession.

Writing a skill on real work is how a team learns it.

Source: LinkedIn, 2024 Workplace Learning Report



Write it down: the launch-brief skill

Acme Inc repo

acme-checkout/

.agents/

agent-memory/ 1 file

agents/ 1 file

rules/ 3 files

● settings.json

skills/

billing-change/ 2 files

launch-brief/

● SKILL.md

● checklist.md

release/ 1 file

● AGENTS.md

db/

docs/

exports/

services/

web/

.agents/skills/launch-brief/SKILL.md

1 ---

2 description: Write a launch brief from a

3 request. Use for any new plan or price.

4 argument-hint: [request link]

5 ---

6 1. Read AGENTS.md and docs/decisions/.

7 2. Name the customer, problem and goal.

8 3. List what changes for support and sales.

9 4. Run checklist.md before you hand it over.

10 One page. A PM and an engineer start from it.

.agents/skills/launch-brief/checklist.md

1 # Before you hand it over

2 - [] Prices match plans.ts

3 - [] Legal wording copied, not rewritten

4 - [] Support has the answers a week early

Owner: product lead

Every brief answers the same questions, whoever writes it.

Who The person who does the task best

How long One real task, then one fix

Pause point Nobody else runs the skill until it has passed one review.

Keep it current

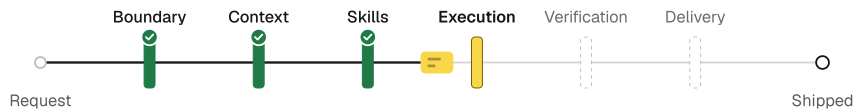
☐ **After each review** Fix the skill with what the review found.

☐ **Each quarter** Retire the skills nobody ran.

Acme Inc after step 3

Example

Boundary	● ● ● ○	Governed
Context	● ● ● ○	Governed
Skills	● ● ● ●	AI-native
Execution	● ● ● ○	Governed
Verification	● ○ ○ ○	Experimenting
Delivery	● ● ○ ○	Assisted



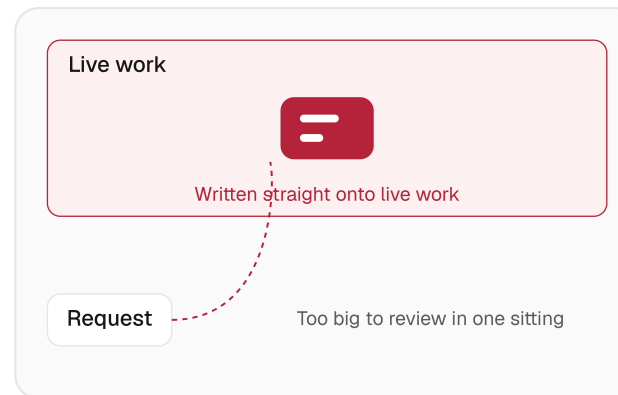
Step 4 of 6 Run agents inside the rules

One big change on live work stalls everything behind it

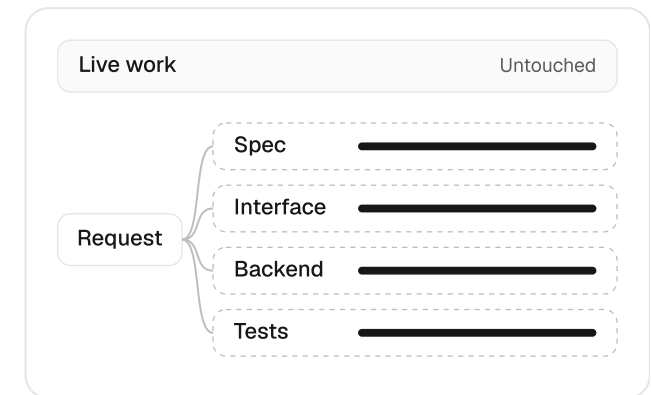
The brief turns into one large change written straight onto live work, and nobody can review it in one sitting. Execution is how work starts from a request and runs in its own space, many tasks at once.

What to do

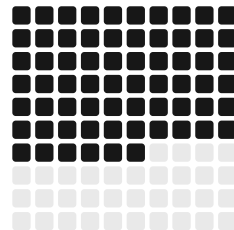
- 1 Start every agent task from a written request with its acceptance checks.
- 2 Give each task its own branch or copy, never the live work.
- 3 Keep changes small enough that one person can review them in minutes.



Today: one large change, written straight onto live work, that nobody can review in one sitting.



With a playbook: the request splits into small tasks, each in its own space, and live work stays untouched.

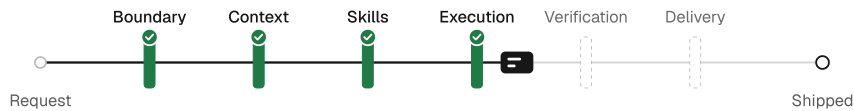


66%

of developers say their biggest frustration is AI solutions that are almost right, but not quite.

Small changes from clear requests are easier to get right.

Source: Stack Overflow, 2025 Developer Survey



Write it down: the billing-change skill

● ● ● Acme Inc repo

acme-checkout/

.agents/

agent-memory/1 file

agents/1 file

rules/3 files

● settings.json

skills/

billing-change/

● SKILL.md

● plan-template.md

launch-brief/2 files

release/1 file

● AGENTS.md

db/

docs/

exports/

services/

web/

.agents/skills/billing-change/SKILL.md

1 ---

2 description: Turn an approved brief into

3 small billing changes.

4 ---

5 1. Explore: read the brief, then the code.

6 2. Plan: one task per ten-minute review.

7 3. Implement each task on its own branch.

8 4. Write each task's test before its code.

9 5. Commit on the branch. Never on main.

10 Stop and ask if a migration is needed.

.agents/skills/billing-change/plan-template.md

1 # Plan: {brief title}

2 ## Tasks, each small enough to review

3 ## Tests written first

4 ## Migrations: none, or ask a person

Owner: engineering lead

Everyone, and every agent, works the same way.

Who

Engineering lead, with one squad

How long

One launch, as a pilot

Pause point

Other squads start only after the pilot ships.

Keep it current

☐ Each change

Keep it small enough to review in minutes.

☐ After the pilot

Write down what the squad changed, then roll out.

Acme Inc after step 4

Example

Boundary

Context

Skills

Execution

Verification

Delivery

● ● ● ○

● ● ● ○

● ● ● ●

● ● ● ●

● ○ ○ ○

● ● ○ ○

Governed

Governed

AI-native

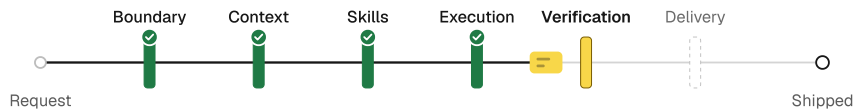
AI-native

Experimenting

Assisted

KaaS.team

15 of 24



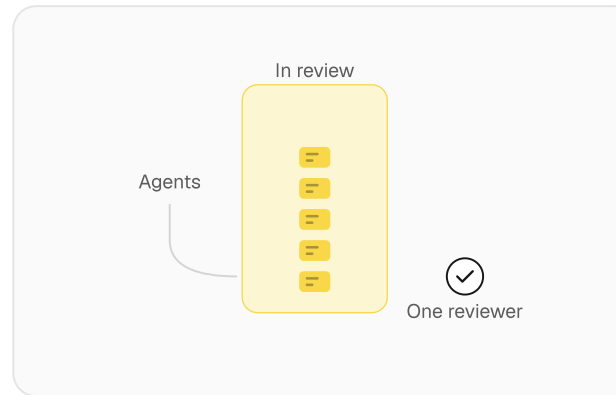
Step 5 of 6 Make done mean the same on every team

Without checks, every change waits for one reviewer

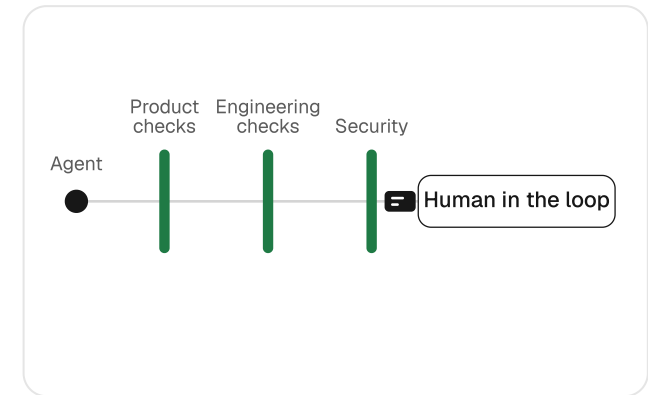
Small changes now arrive faster than one reviewer can read them, and the queue grows. Verification is the checks every piece of work passes before a person reviews it.

What to do

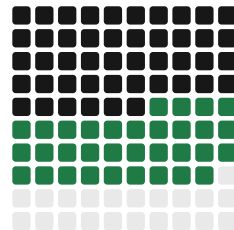
- 1 Write down what done means for code, copy and data, in plain words.
- 2 Turn each line into a check that runs before a person looks.
- 3 Have a separate reviewer agent read each change cold, and send failures back.



Today: changes arrive faster than one reviewer can read them, and the queue grows.



With written checks: product, engineering and security checks run first, and a person sees only what passed.



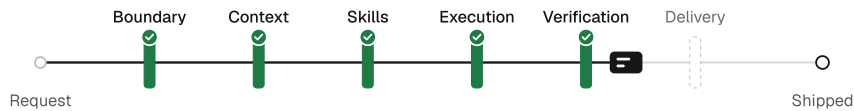
46%

of developers distrust the accuracy of AI tools.

33% trust it.

Checks earn trust one passed change at a time.

Source: Stack Overflow, 2025 Developer Survey



Write it down: the reviewer agent

Acme Inc repo

acme-checkout/

.agents/

agent-memory/ 1 file

agents/

reviewer.md

rules/

billing-service.md

customer-data.md

testing.md

settings.json

skills/ 5 files

AGENTS.md

db/

docs/

exports/

services/

web/

.agents/agents/reviewer.md

```

1 ---
2 name: reviewer
3 description: Reviews each billing change
4   before a person does.
5 tools: Read, Grep, Glob, Bash
6 ---
7 You did not write this diff. Read it cold.
8 Check proration, currency and refunds.
9 Run npm run test:billing and npm run lint.
10 Send it back if a rule in testing.md fails.

```

.agents/rules/testing.md

```

1 ---
2 paths: ["services/**/*.test.ts"]
3 ---
4 Every bug fix ships with the test that
5 would have caught it.

```

Owner: team leads

Reviewers read a change that already passed.

Who

Team leads, with the product lead

How long

One afternoon

Pause point

Loosen review only for checks that pass for two weeks.

Keep it current

☐ **Each week** Track review wait and rework, so the change shows.
 ☐ **A bug reaches users** Add the check that would have caught it.

Acme Inc after step 5

Example

Boundary

● ● ● ○

Governed

Context

● ● ● ○

Governed

Skills

● ● ● ●

AI-native

Execution

● ● ● ●

AI-native

Verification

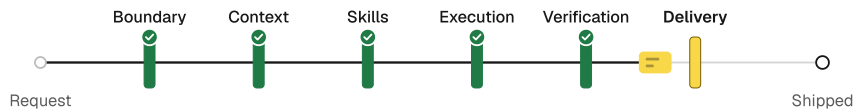
● ● ● ○

Assisted

Delivery

● ● ● ○

Assisted



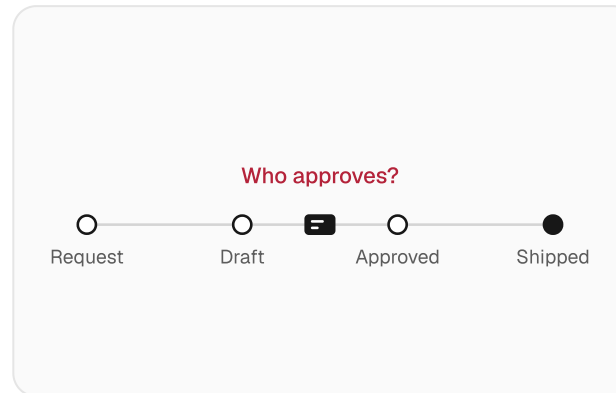
Step 6 of 6 Ship through one path, with a named owner

Fast code still waits when nobody owns the release

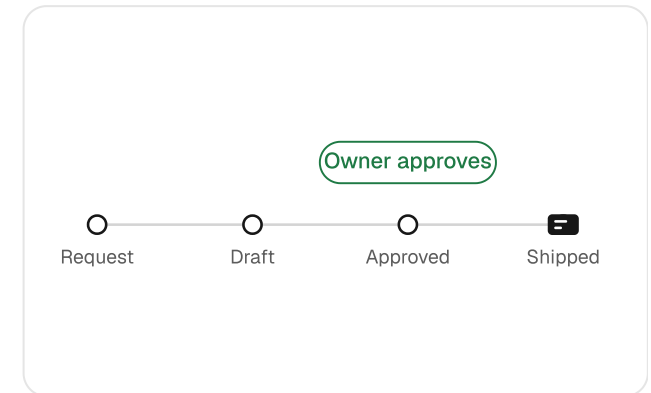
The change passed every check and still waits, because nobody knows who signs off on billing. Delivery is one path from request to shipped, with a named owner approving.

What to do

- 1 Draw the one path every change takes from request to release.
- 2 Name the approver for each kind of change.
- 3 Mark a layer as owned when a team runs it without help.



Today: the change is ready, but it sits between draft and approved because nobody knows who signs off.



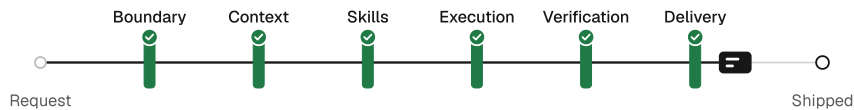
With one path: every change takes the same route, and a named owner approves it before it ships.

Amplifier

is how the research describes AI's main role in software delivery: it magnifies the strengths of high-performing organizations and the dysfunctions of struggling ones.

One clear path is the strength worth amplifying.

Source: DORA, 2025 State of AI-assisted Software Development



Write it down: the release skill

Acme Inc repo

acme-checkout/

.agents/

agent-memory/

reviewer/

● MEMORY.md

agents/ 1 file

rules/ 3 files

● settings.json

skills/

billing-change/ 2 files

launch-brief/ 2 files

release/

● SKILL.md

● AGENTS.md

db/

docs/

exports/

services/

web/

.agents/skills/release/SKILL.md

1 ---

2 description: Ship an approved change.

3 disable-model-invocation: true

4 ---

5 Only a person runs this. Agents cannot.

6 1. Check the reviewer passed the change.

7 2. Billing: the head of product approves.

8 3. Anything else: the engineering manager.

9 4. Release to 10% of accounts, then all.

10 5. Watch the first hour. Roll back on errors.

.agents/agent-memory/reviewer/MEMORY.md

1 # What the reviewer has learned

2 Upgrades to annual charged twice. Check it.

3 Invoice numbers skipped once. Check it.

Owner: engineering manager

Approval is a step in the path, not a message in a chat.

Who

Engineering manager

How long

One meeting

Pause point

A layer counts as owned only when a team runs it without help.

Keep it current

☐

Each release

A person runs the release skill. No agent can start it.

☐

Tools change

Review the files the same quarter.

Acme Inc after step 6

Example

Boundary

● ● ● ○

Governed

Context

● ● ● ○

Governed

Skills

● ● ● ●

AI-native

Execution

● ● ● ●

AI-native

Verification

● ● ○ ○

Assisted

Delivery

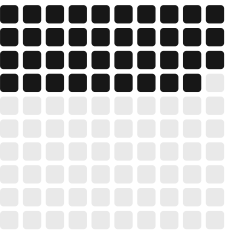
● ● ● ●

Governed

Make it stick: train the people who keep it running

Training sticks when your leads write the files

The files stay useful only if people on your side can write and change them. Training works best on your own repo, on a launch you already planned.



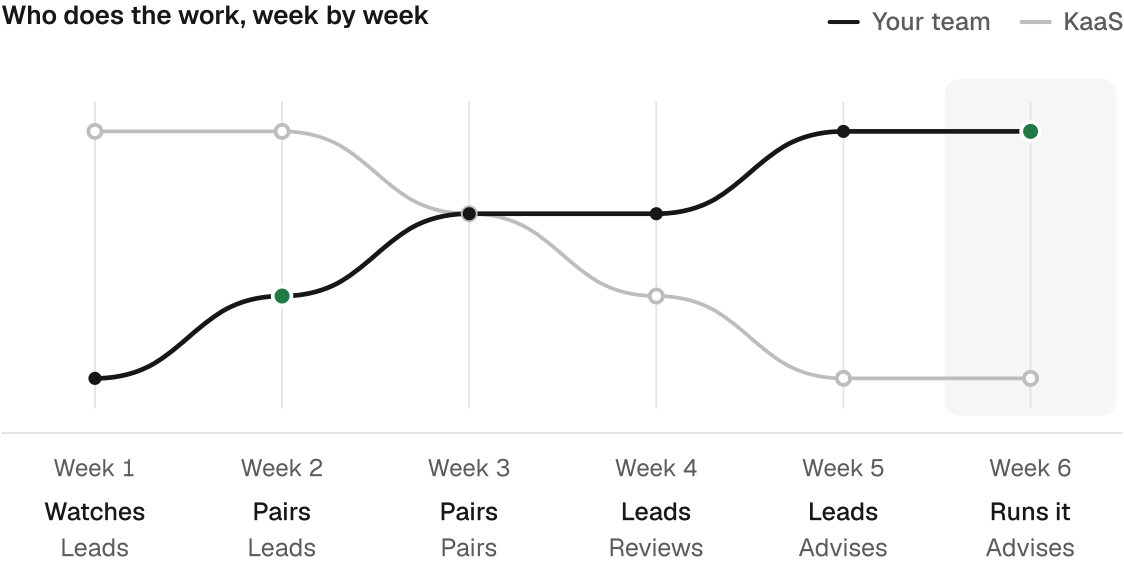
39%
of people who use AI at work have had AI training from their company.

The other 61 in every 100 got none from their employer.

Source: Microsoft and LinkedIn, 2024 Work Trend Index

Your leads write the rules, skills and checks for their own teams, on a launch you already planned. By week six the team runs it, and the files stay with you.

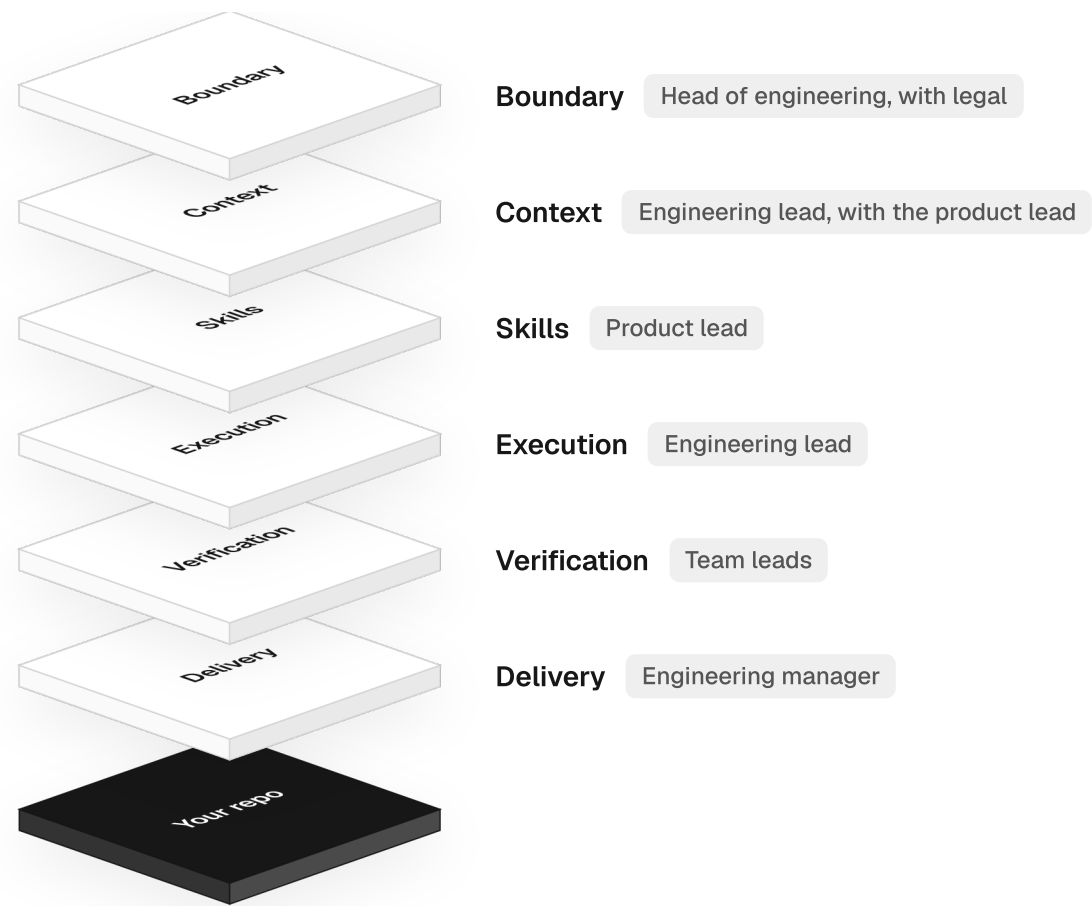
Who does the work, week by week



- **Week 2** Your launch planned with the factory
- **Week 6** Your team ships it, and your approver signs off

Every layer has one named owner

When someone leaves, the files stay, and so does the way of working. A new person, or a new tool, starts from what is written.



Questions leads ask

Short answers to what product and engineering leaders ask before they train a team this way.

What is an AI-ready team?

A team where the rules, context, skills and checks for AI work are written down in the repo, owned by named people, and used by everyone.

Who should be trained first?

The leads who will own the files: a product lead and an engineering lead from one pilot squad.

What does open mean here?

Every rule, skill and check is a readable file. Your teams copy it, change it and review it like code.

Do we need new tools?

No. The files are plain text in your repo and work with the tools your teams already use. They survive a switch.

Is this only for engineers?

No. PMs write the product context and the product checks, and use the same skills for specs and briefs.

How do we know it worked?

Take a baseline of cycle time, review wait and rework before you start, and report against it.

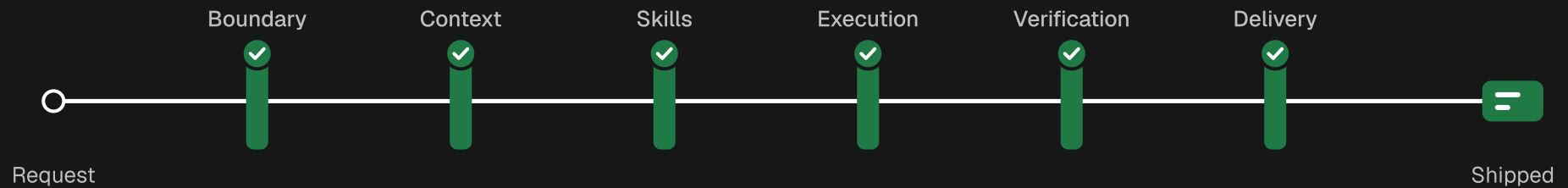
Terms

The terms this playbook uses, each in one sentence.

 Boundary	What each agent may see and change, written down and approved by a person.	 Context	The knowledge each task needs, kept in one place someone maintains.
 Skills	How your best people work, written down so every agent works that way.	 Execution	Work starts from a request and runs in its own space, many tasks at once.
 Verification	Checks every piece of work passes before human review.	 Delivery	A clear path from request to shipped, with a named owner approving.
● ○ ○ ○ Experimenting	People try AI on their own.	● ● ○ ○ Assisted	Daily use, no shared rules.
● ● ● ○ Governed	Shared rules, named owners.	● ● ● ● AI-native	Runs on AI, kept current.
AGENTS.md	The one short file every agent reads first: commands, house style and gotchas. Any agent tool reads it, or a copy under its own name.	Path rule	A rule file that applies only to the folders named in its paths line.
Reviewer agent	A separate agent that reads each change cold, with read and run tools only, before a person does.	Agent memory	What an agent learned on past changes, kept as a file the team can read and correct.

Train your team on your own repo

A custom training for your product and engineering teams, run on a launch you already planned. Your leads write the rules, skills and checks, and your team ships the launch the new way.



Where is your team today?

☐ Experimenting

☐ Assisted

☐ Governed

☐ AI-native

Take the two-minute assessment

Book a call

Send an enquiry

Training only your leads? See the cohort