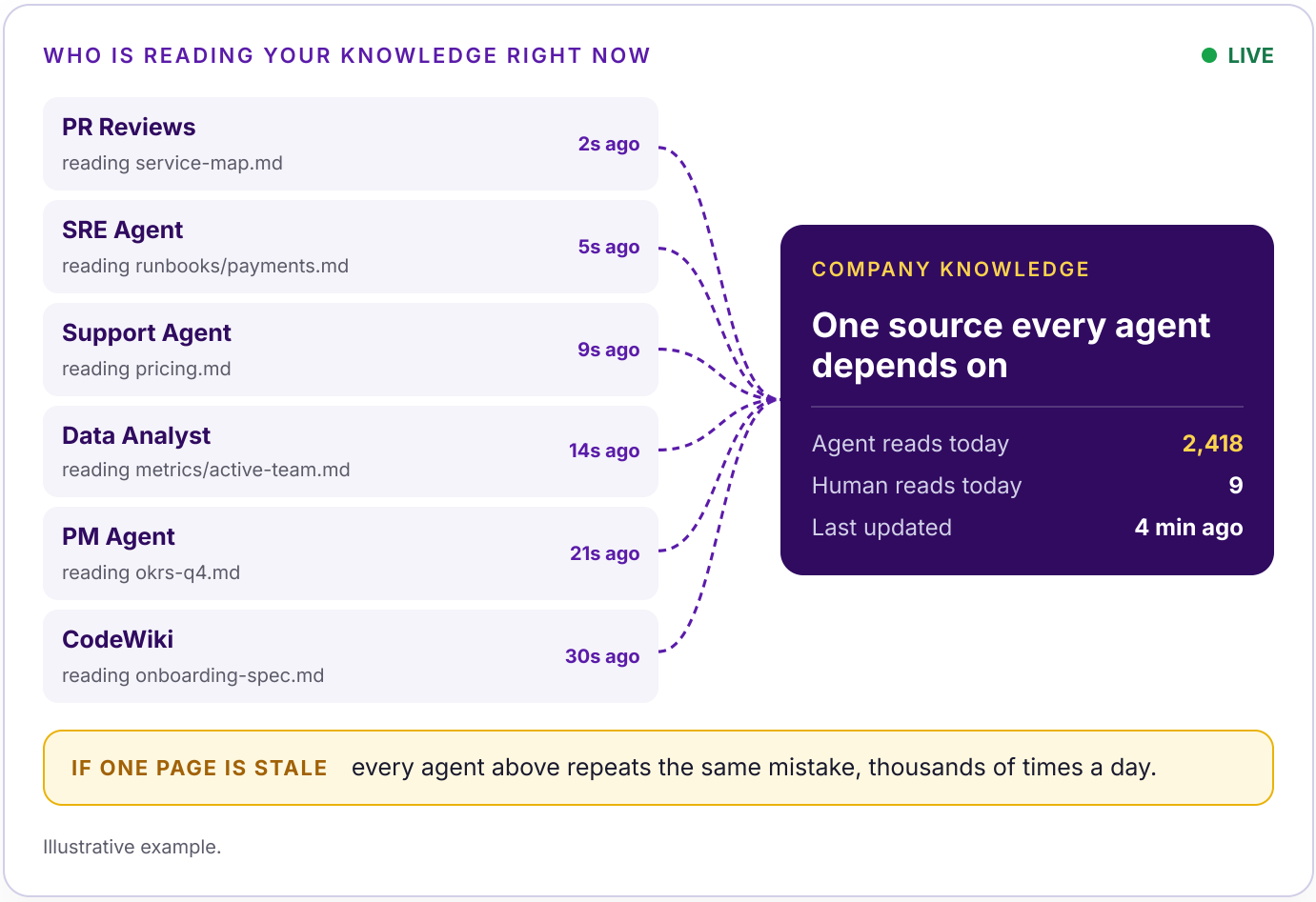
FOR CTOS AND ENGINEERING LEADERS

Your knowledge base is now production code.

Agents read your company's knowledge all day, far more often than your people do. It has to be current every minute. Treat it like code.

Agents need context. Documentation is context.



Illustrative example.

Versioned

Every page has a history

Reviewed

Every change gets a second look

Kept current

Agents do the upkeep

01 · THE DECAY CYCLE

Every wiki follows the same path to irrelevance.

A page is written once and read months later. In between, the product ships new features, the team reorganizes, and the plan moves. The page stays where it was. The people who could correct it are the ones with the least time, and checking old pages belongs to nobody.

DAY 0
Page written
Accurate on the day it is published

MONTH 2
Product ships
Three releases change the behavior

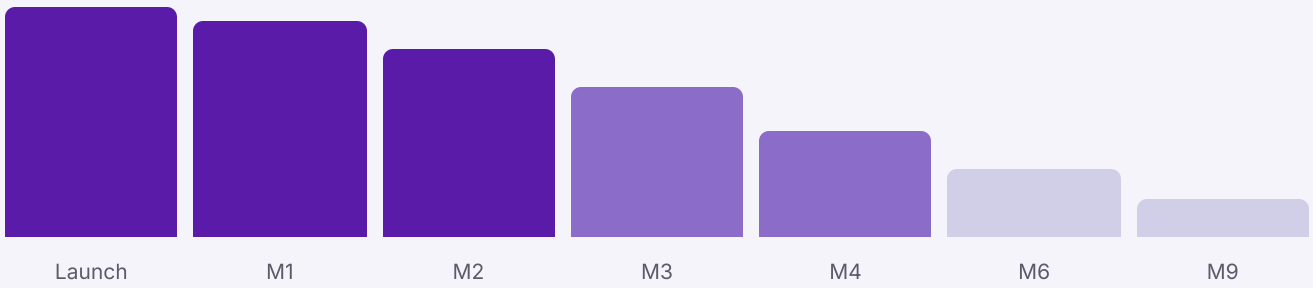
MONTH 4
Team changes
The author moves to another team

MONTH 6
Someone follows it
A coworker says it changed long ago

MONTH 9
Trust is gone
People ask in Slack instead

Trust in the wiki over time

Illustrative



Now a new reader has arrived

A PERSON READS A STALE PAGE

Cross-checks it against what they remember, what they heard in standup, and who to ask.

AN AGENT READS A STALE PAGE

Can only compare it to other pages and threads. When they disagree, it has to guess which to trust, and a capable model guesses with confidence.

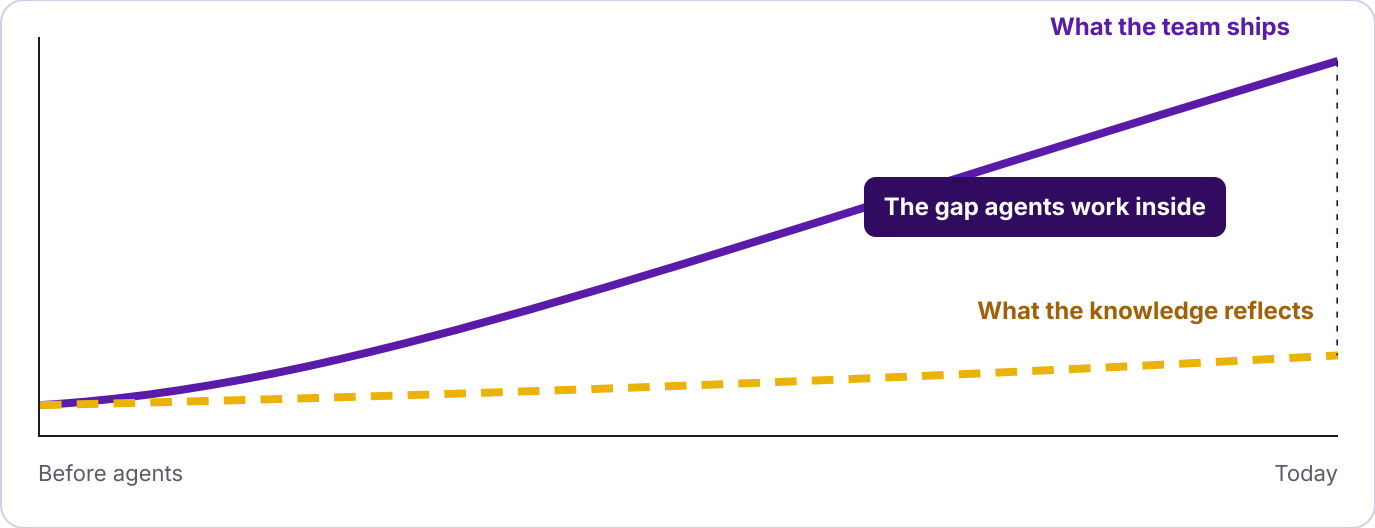
THE SHIFT

A stale page used to mislead a handful of people. Now it misleads every agent that reads it, at machine speed.

02 · THE VELOCITY GAP

The work got faster. The knowledge did not.

Agents have multiplied what small teams ship. The documents those agents depend on, the strategy, the specs, the OKRs, the runbooks, still move at the speed of whoever has time to edit them. The gap widens every sprint.



FROM THE FIELD

A product leader at a large enterprise software company runs a team of 15 that agents have made several times more productive. Asked whether the rest of the organization can keep up, his answer was **“1000% yes”** it is hard.

He will not act on an agent's recommendation until it understands his product strategy and OKRs. Those are exactly the pages that change slowest.

75%+

of traffic to hosted public documentation now comes from agents, not people, according to Mintlify. Internal knowledge is heading the same way.

FOR THE CTO

Your agents are only as current as the slowest document they read. Shipping speed is now capped by knowledge freshness.

03 · THE WRONG FIX

Better search finds the page. It does not make it true.

Most knowledge tools are responding to agents by adding AI on top: an answer sidebar, a summarize button, a connector agents can query. Each one makes retrieval easier. None of them changes who writes the page, where it lives, or who remembers to update it. That was always the part that failed.

AI ON TOP OF THE WIKI

Faster access to the same stale page

- Answer sidebar
- Summarize button
- Search connector for agents

The page underneath is still months old.

vs

KNOWLEDGE AS CODE

A page that updates when the business does

- Agents draft the change
- People review it
- History makes it reversible

The page is correct before anyone searches for it.

Where each approach spends its effort

	SEARCH-FIRST TOOLS	KNOWLEDGE AS CODE
Problem solved	Finding the page	Keeping the page true
Who writes updates	Whoever remembers	Agents draft, people approve
When updates happen	When someone notices	When the source changes
Conflicting pages	Agent picks one	Flagged as a contradiction
Mistakes	Hard to trace	Diff, author, and one-click revert

THE POINT

Finding the page was never the hard part. Keeping it true is.

Engineering already solved this. For code.

Software teams have a proven system for files that machines depend on: every change is tracked, reviewed, versioned, and easy to undo. Company knowledge now has the same kind of reader, so it deserves the same discipline. The simplest version keeps knowledge as plain files in Git, owned by you, readable by people and agents from the same place.

COMPANY-KNOWLEDGE/

strategy/	
2027-plan.md	2d ago
okrs-q4.md	5d ago
product/	
onboarding-spec.md	today
pricing.md	today
engineering/	
service-map.md	1h ago
runbooks/	
sales/	
playbook.md	flagged

Plain files, full history, one source for people and agents.

01 **Version history.** Every page shows who changed what, when, and why.

02 **Review.** Changes are proposed and approved, like a pull request.

03 **Diffs.** Reviewers see exactly what moved, not a whole new page.

04 **Revert.** A wrong update is one click from being undone.

05 **Checks.** Contradictions between pages are caught before they merge.

Structure pays for itself

+64%
more precise answers when agents worked from structured docs instead of raw code

~50%
fewer tokens used for the same questions

Source: controlled test with a large engineering team using a coding agent, reported by Mintlify.

People decide what is true. Agents do the typing.

Writing company knowledge should feel like working with a coding agent. It drafts alongside you, pulls in the context you forgot, asks what you meant, and points out where the new text contradicts an existing page. Then it keeps working after you close the tab, turning new source material into updates and flagging gaps.

A pricing change, end to end

FRI 16:00
Change ships
New enterprise pricing goes live in the product.

FRI 16:05
Agent detects it
Finds every page that still states the old price.

SAT–SUN
Drafts and flags
Proposes fixes on 4 pages and flags a sales playbook that now conflicts.

MON 11:30
Approved
A sales lead reviews the diffs and merges them before lunch.

Draft
Agents write the first version of new pages from specs, tickets, and threads.

Maintain
Agents watch sources and propose updates when the truth changes.

Answer and fix
An agent answering a question can also correct the page it used.

AGENTS OWN

- Drafting and rewriting
- Finding affected pages
- Spotting contradictions and gaps

PEOPLE OWN

- **What is true**
- **Approving every change**
- **Strategy and priorities**

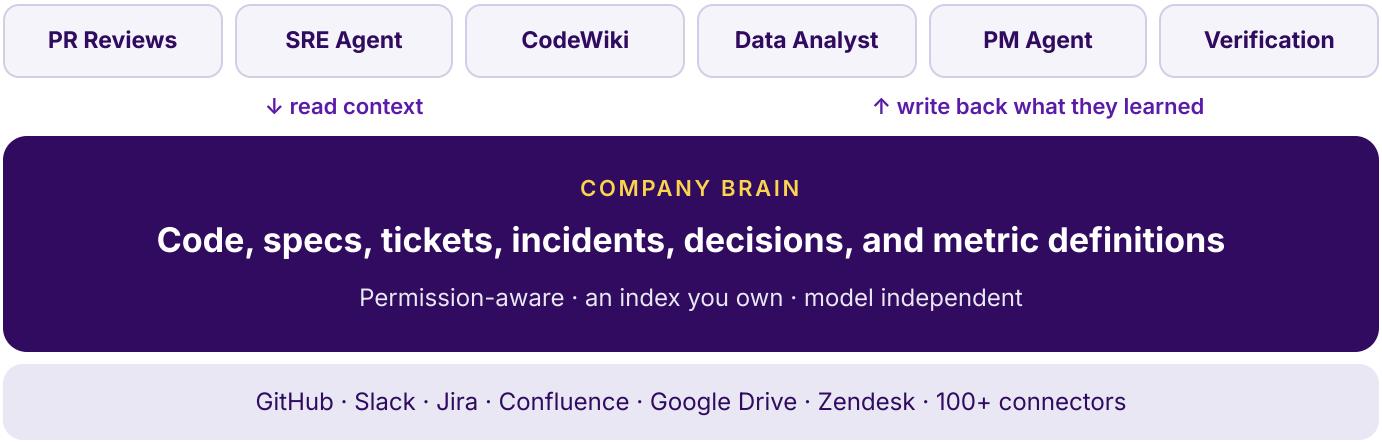
EXPECT THIS SOON

Within the next two years, agents will handle most knowledge upkeep at most companies, with people reviewing changes the way engineers review code.

06 · HOW XHAWK APPROACHES IT

A Company Brain that every AI employee reads and writes.

XHawk treats company knowledge as shared infrastructure for its AI employees. The Company Brain indexes the sources your teams already use, and every agent that works from it writes back what it learned. Knowledge stops being a side project and becomes part of how work gets done.



Knowledge that stays current as a by-product of work

CodeWiki

Answers from the current code, so engineering knowledge never trails the repository.

SRE Agent

Drafts each postmortem and adds it to the Brain, so the next incident starts with history.

PR Reviews

Learns from human review comments and carries those standards to the next pull request.

Data Analyst

Codifies metric definitions once, so every dashboard and agent uses the same number.

THE DIFFERENCE

Most tools ask people to maintain knowledge so agents can use it. XHawk AI employees maintain it as they work.

Five questions to ask your team this week.

01	When did someone last verify your product strategy page?
02	Which documents do your agents read most, and who owns them?
03	If a policy changes on Friday, how many pages are wrong on Monday?
04	Can you see who changed a page, why, and undo it?
05	Do your agents and your people work from the same source?

01

Start where agents read most

Specs, runbooks, and service docs are the highest-leverage pages.

02

Move them into version control

History, review, and revert from day one.

03

Let agents propose, people approve

Upkeep becomes a review queue, not a chore.

Your wiki is becoming a codebase. Run it like one.

One Company Brain. Always on. AI Employees for Product, Engineering & Analytics.

Learn more about CodeWiki from XHawk →

© 2026 XHawk. All rights reserved.