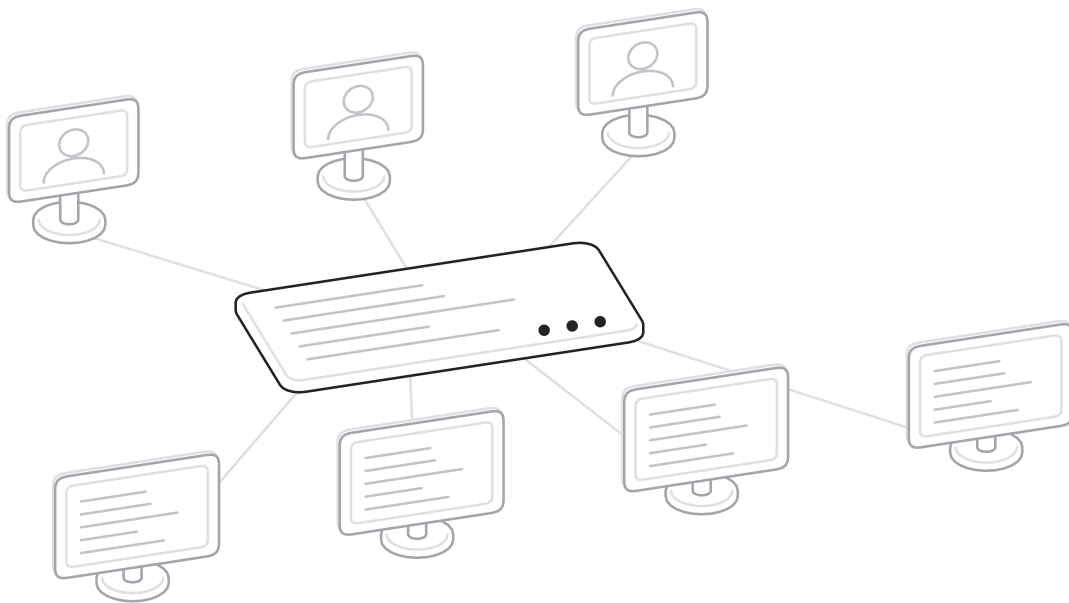


GUIDE

Before the agents start

How engineering teams agree on what to build, then run more of it at once.

For engineering leaders whose teams already use Claude Code, Cursor, or Codex.



Contents

	In short	3
01	Building got cheap. Agreeing did not.	4
02	Work one level up.	5
03	Intent belongs to the team.	5
04	Write the Brief.	6
05	Agree once, by name.	8
06	Capture decisions as the work happens.	9
07	One live source of truth for every agent.	10
08	Direction, discovery, delivery, at the same time.	11
09	Map what is running.	12
10	How do you show that AI is working?	13
11	A 30-day rollout.	14
12	Where Hamster fits.	15
	Common questions	15
	Sources	16

Web version, with the templates to copy: tryhamster.com/guides/before-the-agents

In short

Intent-driven development is a practice in which a team runs direction, discovery, and delivery as connected loops on one shared record of its intent, decisions, and knowledge, so agents build what the team agreed and each person directs more work at once.

-
- Writing and testing code takes only about a quarter to a third of the time from idea to launch, according to Bain & Company's 2025 Technology Report, so faster code alone leaves most of the timeline unchanged.
-
- On teams with high AI adoption, developers merged 98% more pull requests while time spent in review rose 91%, according to Faros AI's 2025 telemetry from more than 10,000 developers.
-
- Before any agent starts, the team writes a one-page Brief with five sections (Context, Goals, Phases / Approach, Scope, and Next Steps), and everyone whose work it changes votes Ready or Not yet.
-
- Every agent session reads the current Brief and decision log before each step and writes its decisions back while it works, so the record compounds as the team ships.
-
- Engineering leaders can show that AI is working by counting what the team directs, such as concurrent agreed workstreams per person and rework, instead of token spend, lines of code, or suggestion acceptance.
-

Your team already runs coding agents. Claude Code, Cursor, and Codex write in an afternoon what used to take a sprint, and most engineers on most teams have at least one of them open right now. The hard part has moved. It now sits in the minutes before an agent starts, when somebody decides what it should build, and in the hours after, when the team finds out whether everyone meant the same thing.

This guide is about that part. It is written for the person who answers for the team's output: an engineering manager, a director, a VP, a CTO. It describes a practice we call intent-driven development, in which a team runs direction, discovery, and delivery as connected loops on one shared record of its intent, decisions, and knowledge, so agents build what the team agreed and

each person directs more work at once. It ends with six templates you can use this week, and a version of the whole guide your coding agent can run with you.

01

Building got cheap. Agreeing did not.

Nine in ten software professionals now use AI at work, a measure of how fast the tools spread.¹ Shipping has sped up less than the demos suggest, because the part that got faster is a minority of the work. Bain found that writing and testing code takes about a quarter to a third of the time from an idea to a launch.² Speed up that slice tenfold and the rest of the timeline, the deciding, the agreeing, the reviewing, is still there.

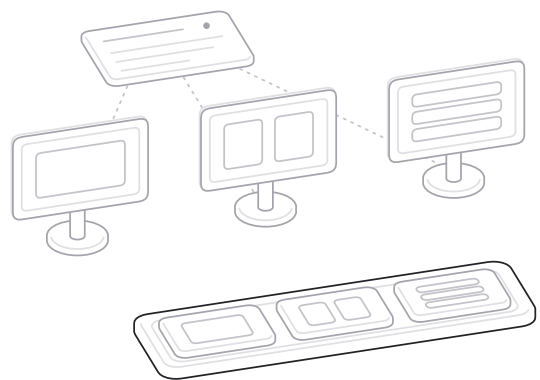
Where agents do speed up the code, the cost reappears downstream. Faros looked at telemetry from more than 10,000 developers and found that on teams with high AI adoption, developers merged 98% more pull requests while time spent in review rose 91%.³

98% more pull requests merged.
91% more time spent in review.

Faros AI, The AI Productivity Paradox, 2025

Review time rises for a reason every engineering leader recognizes. Three people read the same ticket alone, each with a slightly different picture of done. Each opens a session, and each agent faithfully builds that person's picture. Before agents, the gap between those pictures surfaced slowly, in a standup or a design review. Now it surfaces all at once, as three finished pull requests, and the team pays for each one in review and rework.

Google's DORA team put the general rule plainly: "AI's primary role in software development is that of an amplifier. It magnifies the strengths of high-performing organizations and the dysfunctions of struggling ones."¹ With agents, a team that agrees badly gets its disagreement built into code.



02

Work one level up.

Agents now carry whole tasks. When Anthropic studied how people use Claude Code, it classified 79% of conversations as automation, where the agent did the task, rather than augmentation, where it helped a person do it.⁴ Anthropic's 2026 report adds the other half of the picture: developers use AI in roughly 60% of their work, yet say they can "fully delegate" only 0 to 20% of their tasks.⁵

Read together, the two numbers describe the new job. Agents execute. People still decide what should be executed, check that it was the right thing, and carry the context that makes the next task go well. The leverage in that arrangement comes from how well a person directs, not how fast they type.

Picture two engineers with the same tools. The first opens a session, explains the feature, pastes in the constraints, corrects the agent twice, and ships one workstream by Friday. The second writes down the intent once, gets the two people it affects to agree, and hands the agreed version to agents in three sessions while she reviews and decides. She runs three workstreams to the first engineer's one, because she works one level up.

So count a team's output differently. The useful unit is the number of agreed workstreams the team can run at once, from direction through discovery to delivery, without losing track of why each one exists.

03

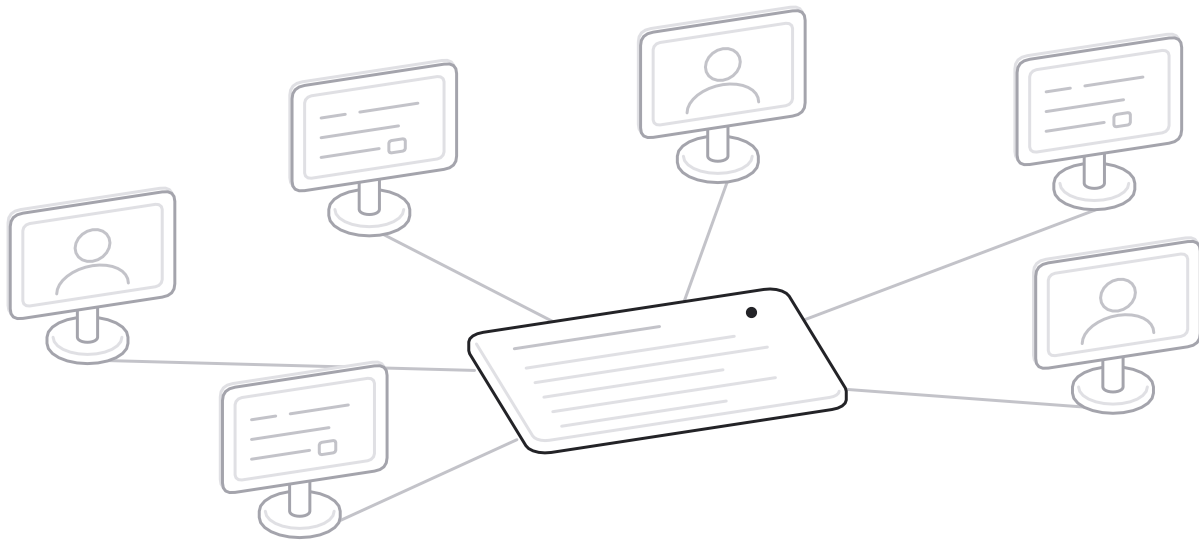
Intent belongs to the team.

The most common frustration developers report with AI tools is output that is "almost right, but not quite." Two in three name it as their biggest one.⁶ Some of that gap is the model. Much of it is intent the agent never received: the constraint someone knew, the case someone had decided to leave out, the reason the obvious design was rejected last month.

A single-author spec does not close that gap, because it is written by one person and read by one agent. Team intent is different in four ways:

1. Several people shape it.
2. The people whose work it changes agree to it by name.
3. The decisions behind it are kept, with who made them.
4. Every agent session works from the same version, and sees it change.

The fourth point matters more as teams grow. Fred Brooks counted the communication channels in a team of n people as $n(n-1)/2$.⁷ Five people have ten channels between them; ten people have forty-five. Give each of those people several agent sessions and the channels multiply again, because every session is another reader who needs the same context. Telling each one separately stops working quickly. The intent has to live in one place that every person and every agent reads from.



In Hamster that place is the Brief. A Brief is not a PRD. It is a multiplayer conduit for a team's intent: the people involved shape it together, vote on it, and argue in it, and agents execute from it. The rest of this guide shows the practice with plain templates, so you can run it with or without Hamster.

04

Write the Brief.

Before any agent starts, write one page. In Hamster that page is a Brief, and it has the same shape every time: a title and a one-line description in their own fields, then five short sections. It is a product document, about what and why. Technical direction goes on the Brief's companion Spec, so the Brief stays readable by everyone whose work it changes.

TEMPLATE **Brief**

Context

The problem or need, who it affects, and why it matters now.

Goals

- What this work should achieve, written as outcomes someone else can check.

Phases / Approach

- The approach at the level of outcomes and sequence. No file names, components or code.

Scope**In scope**

- ...

Out of scope

- ...

Next Steps

- Open questions and decisions still needed, each with an owner.

Context says what is broken or missing from the user's side before it says anything about a solution. Agents optimize for what you describe; describe the problem and they can weigh two designs that both work.

Goals must be checkable by someone who did not write them. "Admins can remove a user in one step" survives contact with an agent better than "improve user management." If the author is the only person who can tell whether the work is finished, the team will discover the real criteria in review.

Phases / Approach describes the order of outcomes, not the implementation. Architecture, data and stack choices belong on the Spec, where each technical decision is recorded with its context, the options considered and the consequences.

Scope does more work than any other section, and its out-of-scope half does the most. Agents are eager. Without a written edge they will helpfully refactor the module next door, and the reviewer has to decide whether that was wanted.

Next Steps is where open questions live, each with an owner. An empty section is useful information too: it marks a conversation the team has not had yet, and it is far cheaper to have it now than after three agents have built three answers to it.

There is no "agreed by" line to fill in. In Hamster, each person whose work the Brief changes votes Ready or Not yet, and a Not yet vote needs a comment. Each person's latest vote counts, and the Brief's Activity tab keeps the history.

Try it with your AI

Your coding agent can run this guide with you. Paste one line into Claude Code, Cursor, or Codex, in the repo your team works in.

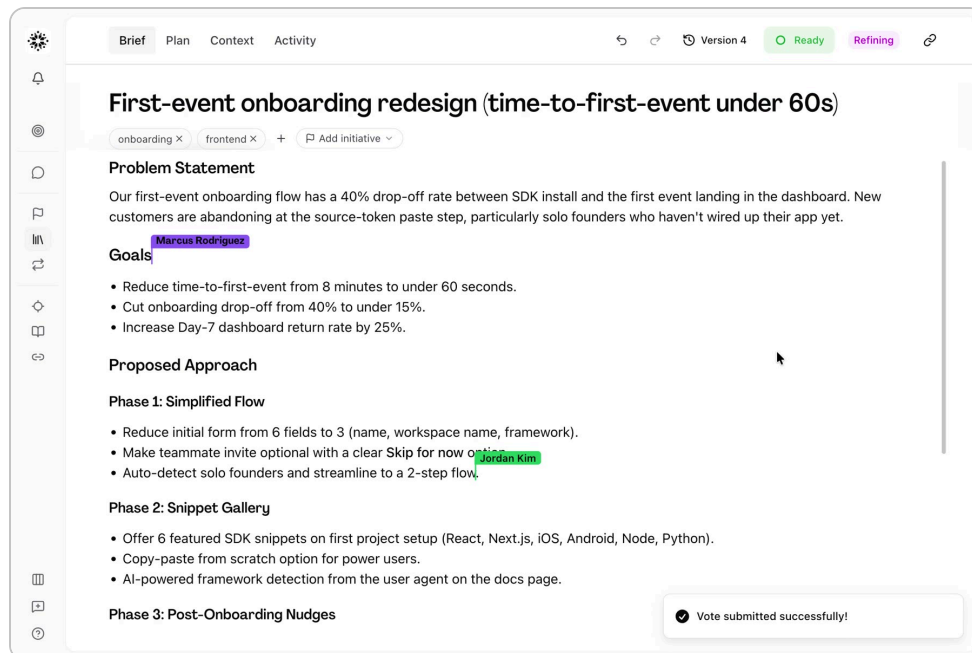
Read <https://tryhamster.com/guides/before-the-agents/skill.md> and run it with me.

1. It connects Hamster. You sign in once in the browser; new readers create a free account there.
2. It drafts the Brief from your repo and asks only about the gaps.
3. It creates the Brief, starts the decision log, and keeps writing decisions back as it works.

05

Agree once, by name.

Alignment should be a short, explicit step, not a meeting. The people whose work the Brief changes read it and vote Ready or Not yet. Every Not yet vote comes with a written concern and an owner. Concerns are resolved in the open, where the next reader can see how they were settled. Agents build from the Brief the team agreed to, so adopt a team rule that everyone confirms their vote again after a material change.



Rework is the bill for skipping this step, and it was large before agents arrived: Boehm and Basili's widely cited estimate put avoidable rework at 40 to 50% of a software project's effort.⁸ Agents make rework faster to produce and easier to hide. GitClear's analysis of changed code

found copy-pasted lines rising from 8.3% to 12.3% between 2021 and 2024, while moved lines, a sign of code being reorganized and reused, fell from 25% to under 10%. 2024 was the first year in its data that pasted code outweighed moved code.⁹ Code written from a shared understanding tends to reuse what exists. Code written from five private understandings tends to repeat it.

Run this checklist before the Brief goes to agents.

TEMPLATE **Alignment checklist**

- ☐ Everyone whose work changes has read the current Brief.
- ☐ Each of them has voted Ready or Not yet.
- ☐ Every Not yet vote has a written concern and an owner.
- ☐ Everyone has read the Out of scope list and raised what they disagree with.
- ☐ Each Next Step that blocks delivery has an answer.
- ☐ Each other Next Step has an owner.
- ☐ Someone who did not write the Goals can check each one.
- ☐ Each technical constraint is on the Spec.
- ☐ Each technical constraint names the files, services, or standards that it applies to.
- ☐ The decision log has an entry for every choice made while writing this.
- ☐ One person owns the outcome after agents deliver.
- ☐ After a material change, everyone confirms their vote again.

06

Capture decisions as the work happens.

Intent and decisions both matter, and a team cannot share a decision that was never captured. Capture each decision against the intent it serves, as the work is done, not afterward in a document nobody has time to write.

Teams have known for a long time that decisions need a written home. Michael Nygard proposed architecture decision records in 2011: a title, the context, the decision, its status, and its consequences, kept beside the code.¹⁰ Thoughtworks moved lightweight decision records to Adopt on its Technology Radar in 2018.¹¹ Most teams still lose most of their decisions anyway, because a decision record is written after the fact, by someone who has already moved on. Half of developers say they lose ten or more hours a week to work outside coding, and finding information is one of the largest drains.¹²

Agents change the economics in both directions. They make dozens of small decisions in every session, about structure, naming, edge cases, and which of two approaches to take, and they forget all of them when the session ends. They are also the cheapest scribes a team has ever had. A session that is told to write its decisions back as it works will do it reliably.

Intent-driven development is built on that habit: as your sessions work, they write to a durable, shared record of the team's decisions, context, and intent. The record builds and compounds as you ship. Each new piece of work starts from everything the team has already decided, so it starts further ahead, and the leverage of every person grows over time.

Keep the record simple enough that a person or an agent can add a line in seconds.

TEMPLATE Decision log					
Date	Decision	Alternatives rejected	Decided by	Reversible?	Workstreams affected

"Decided by" can name an agent session, as long as a person confirms the decision.

"Alternatives rejected" is the field teams skip and later miss most, because it is what stops the next session from proposing the rejected design again.

We built a full workspace to make intent-driven development possible for our own team, and our leverage has skyrocketed since.

Eyal Toledano, founder of Hamster

07

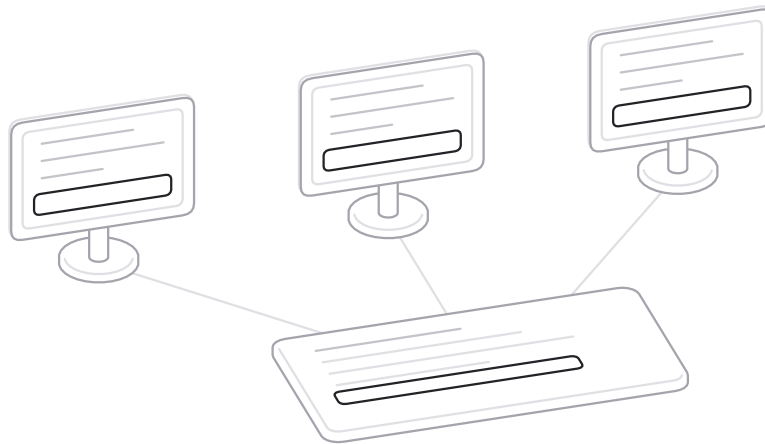
One live source of truth for every agent.

Teams now run Claude Code, Cursor, Codex, and cloud agents side by side, often on the same feature. If each agent gets its context from whoever opened the session, each one builds a slightly different product.

The industry has started to standardize the plumbing. [AGENTS.md](#), a plain file of instructions for coding agents, had been adopted by more than 60,000 open-source projects and agent frameworks by December 2025, when OpenAI contributed it to the Linux Foundation's new Agentic AI Foundation alongside Anthropic's Model Context Protocol and Block's [goose](#).¹³ Anthropic reported more than 10,000 active public MCP servers and more than 97 million monthly SDK downloads at the time.¹⁴ DORA lists AI-accessible internal data among the seven capabilities that decide whether AI helps an organization or hurts it.¹⁵

A file in the repo and a connection to a server are necessary. They are not sufficient, because a team's intent is not static. It has to reach agents in real time. As the team decides things, sessions that are already running should build from the new decision too, not only the sessions that start tomorrow. A session that read the Brief at nine o'clock and never looked again is working from a version the team has since changed.

So the source of truth has three properties. Every agent reads from it, through the repo or [MCP](#). Every agent writes back to it as it works. And it is live: when a person or a session records a decision, the next step of every other session can see it.



08

Direction, discovery, delivery, at the same time.

Once intent is explicit and decisions are captured as they happen, a team can run different stages of work at once. One person sets direction for next quarter while two discovery threads test approaches and three [delivery workstreams](#) build from agreed intent. People decide at the gates between stages, and agents carry the work inside each stage.

The companies getting the most from AI already use it across stages. McKinsey found that top performers were six to seven times more likely than their peers to scale AI to four or more use cases, "from design and coding to testing, deployment, and adoption tracking." Those top

performers reported improvements of 16 to 30% in team productivity, customer experience, and time to market.¹⁶

Concurrency has a known failure mode, and it is the queue. Donald Reinertsen wrote that “few developers realize that queues are the single most important cause of poor product development performance.”¹⁷ With agents, the queue forms in front of the people: decisions waiting for an owner, pull requests waiting for review, intent waiting for agreement. DORA’s research points the same way, finding that working in small batches predicts both software delivery performance and organizational performance.¹⁸ Keep each workstream small, and keep the human decisions moving.

09

Map what is running.

A leader who cannot see every workstream on one page will limit the team to what fits in their head. One row per workstream is enough.

TEMPLATE Concurrency map						
Workstream	Stage (direction / discovery / delivery / review)	Owner	Intent (link, version)	Agents on it	Waiting on	Next human decision

The last two columns carry the map. “Waiting on” shows where the queue is forming. “Next human decision” names the one thing that unblocks the row, and who owes it.

Watch the load on people as concurrency rises. Switching between workstreams has a cost even when it does not show up in throughput. In a study of interrupted work, Gloria Mark and colleagues found that “people compensate for interruptions by working faster, but this comes at a price: experiencing more stress, higher frustration, time pressure and effort.”¹⁹ Add a workstream when the map shows the team has decisions to spare, not when the agents have capacity to spare. Agents almost always have capacity to spare.

How do you show that AI is working?

Count what the team directs rather than what the agents type, and read five measures together each month.

Sooner or later a board member or a CFO asks whether the AI spend is working. Self-reports will not answer them. In METR's randomized trial, experienced open-source developers using early-2025 AI tools took 19% longer to finish their tasks, and afterward believed the tools had made them 20% faster.²⁰ METR's 2026 follow-up estimated a speedup for the returning developers, with a confidence interval wide enough to include no effect at all.²¹ DORA's 2024 data tied a 25% rise in AI adoption to an estimated 1.5% drop in delivery throughput and a 7.2% drop in delivery stability.²²

Developers took 19% longer with AI, and believed afterward they had been 20% faster.

METR, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity," July 2025

Activity numbers will not answer them either. The SPACE framework's authors warned that developer productivity "cannot be measured by a single metric or dimension."²³ And any number you turn into a target bends toward the target. Marilyn Strathern's phrasing of Goodhart's law is the one most people remember: "When a measure becomes a target, it ceases to be a good measure."²⁴

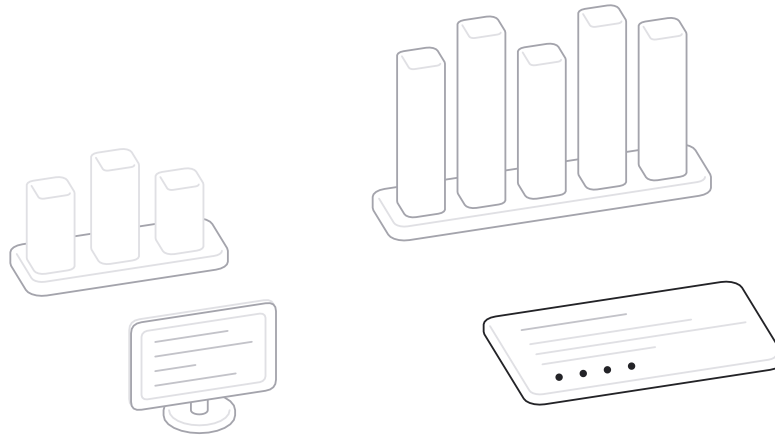
Read the five measures together, and never as a leaderboard.

TEMPLATE **Leader's scorecard**

1. Concurrent workstreams per person running from agreed intent.
2. Share of agent work that traces to an agreed intent.
3. Rework: pull requests rewritten or abandoned because the intent was wrong or missing.
4. Time from intent agreed to merged pull request.
5. Human review time per merged pull request.

Ignore: token spend, lines of code, raw suggestion acceptance.

Token spend, lines of code, and raw suggestion acceptance all rise when nothing improves, which is why they are on the ignore list. The five measures move only when the team directs more work, from better intent, with less rework.



11

A 30-day rollout.

Start with one workstream. Add the second only when the first runs from agreed intent, and read the scorecard before adding a third.

TEMPLATE 30-day rollout

- Week 1: choose one workstream, write its Brief, and start the decision log.
- Week 2: run the alignment checklist, then hand the agreed intent to agents through the repo or MCP. Ask every session to write its decisions back to the log as it works.
- Week 3: add a second workstream in a different stage, and start the concurrency map.
- Week 4: read the scorecard and decide whether to add a third workstream.

Pick the first workstream for its visibility, not its size. A two-week piece of work that several people care about will teach the team more than a quiet refactor, because the Brief and the vote will be tested by real disagreement. By the end of the month you should be able to answer three questions from the record alone: what the team agreed, what it decided along the way, and what it shipped because of both. In Hamster, a scheduled Routine can put the scorecard in front of the team on the same day each month.

Where Hamster fits.

A team that runs many agents at once is running a software factory. Hamster makes that factory build what the team agreed before it builds anything. The Brief is where the team shapes that intent together. Each teammate votes Ready or Not yet, the latest vote from each person counts, and the Activity tab keeps the history. Once the team agrees, Hamster drafts a Plan that agents build from. Decisions and context live in Hamster's Context Graph, Skills and Methods carry the team's ways of working, and agents in Claude Code, Cursor, and Codex read from all of it and write back to it through the Hamster plugin while they work. The record compounds with every piece of work the team ships.

You can run everything in this guide with plain files. If you want the record to stay live across every person and every agent session, Hamster was built for that.

Common questions

How do I keep coding agents aligned with what my team agreed?

Write one page of intent before any agent starts, and have everyone whose work it changes vote Ready or Not yet on it. Then make every agent session read the current version before each step and write its decisions back to a shared decision log.

What is intent-driven development?

Intent-driven development is a practice in which a team runs direction, discovery, and delivery as connected loops on one shared record of its intent, decisions, and knowledge, so agents build what the team agreed and each person directs more work at once. The record compounds as the team ships, so each new piece of work starts from everything already decided.

What should a Brief for coding agents contain?

A Brief has a title, a one-line description, and five sections: Context, Goals, Phases / Approach, Scope, and Next Steps. It is a product document about what and why, and architecture, data, and stack choices go on the Brief's technical Spec.

Why is code review now the bottleneck on teams using AI agents?

Faros AI's 2025 telemetry from more than 10,000 developers found that teams with high AI adoption merged 98% more pull requests while review time rose 91%. When several people brief agents separately, the agents build several versions, and the team pays for each one in review.

How do I measure whether AI coding tools are working for my team?

Self-reports mislead: in METR's 2025 trial, experienced developers took 19% longer with AI tools and believed afterward they had been 20% faster. Read five measures monthly, including concurrent agreed workstreams per person, rework, and review time per merged pull request, and ignore token spend, lines of code, and raw suggestion acceptance.

How should an engineering team roll out coding agents?

Start with one workstream. In week 1 write its Brief and start the decision log, in week 2 run the alignment checklist and hand the agreed intent to agents, add a second workstream in week 3, and read the scorecard in week 4 before adding a third.

Does this work with Claude Code, Cursor, and Codex?

Yes. Agents in Claude Code, Cursor, and Codex read the Brief and write decisions back through the Hamster plugin and MCP server, and the guide's agent skill can draft a team's first Brief from inside the repository.

Sources

1. DORA (Google Cloud), State of AI-assisted Software Development, 2025. <https://dora.dev/research/2025/dora-report/>
2. Bain & Company, "From Pilots to Payoff. Generative AI in Software Development," Technology Report 2025. <https://www.bain.com/insights/from-pilots-to-payoff-generative-ai-in-software-development-technology-report-2025/>
3. Faros AI, The AI Productivity Paradox, 2025. Telemetry from more than 10,000 developers on 1,255 teams; correlational. <https://www.faros.ai/blog/ai-software-engineering>
4. Anthropic, Anthropic Economic Index, "AI's impact on software development," 2025. <https://www.anthropic.com/news/impact-software-development>
5. Anthropic, 2026 Agentic Coding Trends Report. <https://resources.anthropic.com/hubfs/2026%20Agentic%20Coding%20Trends%20Report.pdf>
6. Stack Overflow Developer Survey 2025, AI section. <https://survey.stackoverflow.co/2025/ai>
7. Frederick P. Brooks Jr., The Mythical Man-Month, Addison-Wesley, 1975.

8. Barry Boehm and Victor R. Basili, "Software Defect Reduction Top 10 List," IEEE Computer, 2001. <https://www.cs.umd.edu/~basili/publications/journals/J81.pdf>
9. GitClear, AI Copilot Code Quality, 2025. Share of changed lines, 2021 against 2024. https://www.gitclear.com/ai_assistant_code_quality_2025_research
10. Michael Nygard, "Documenting Architecture Decisions," Cognitect, November 2011. <https://www.cognitect.com/blog/2011/11/15/documenting-architecture-decisions>
11. Thoughtworks Technology Radar, "Lightweight Architecture Decision Records," Adopt, 2018. <https://www.thoughtworks.com/radar/techniques/lightweight-architecture-decision-records>
12. Atlassian, State of Developer Experience Report, 2025. <https://www.atlassian.com/blog/developer/developer-experience-report-2025>
13. Linux Foundation, "Linux Foundation Announces the Formation of the Agentic AI Foundation," December 9, 2025. <https://www.linuxfoundation.org/press/linux-foundation-announces-the-formation-of-the-agentic-ai-foundation>
14. Anthropic, "Donating the Model Context Protocol and establishing the Agentic AI Foundation," December 9, 2025. <https://www.anthropic.com/news/donating-the-model-context-protocol-and-establishing-of-the-agent-ai-foundation>
15. DORA (Google Cloud), the DORA AI Capabilities Model, 2025. <https://cloud.google.com/blog/products/ai-machine-learning/introducing-doras-inaugural-ai-capabilities-model>
16. McKinsey & Company, "Unlocking the value of AI in software development," 2025. <https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/unlocking-the-value-of-ai-in-software-development>
17. Donald G. Reinertsen, The Principles of Product Development Flow, 2009, chapter 1. <http://lpd2.com/wp-content/uploads/2013/06/ReinertsenFLOWChap1.pdf>
18. DORA, "Capabilities. Working in small batches." <https://dora.dev/capabilities/working-in-small-batches/>
19. Gloria Mark, Daniela Gudith, and Ulrich Klocke, "The Cost of Interrupted Work. More Speed and Stress," CHI 2008. <https://ics.uci.edu/~gmark/chi08-mark.pdf>
20. METR, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity," July 2025. <https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/>
21. METR, update to the developer productivity experiment, February 2026. <https://metr.org/blog/2026-02-24-uplift-update/>
22. DORA (Google Cloud), Accelerate State of DevOps Report, 2024. Reported as an association, not a cause. <https://cloud.google.com/blog/products/devops-sre/announcing-the-2024-dora-report>
23. Nicole Forsgren, Margaret-Anne Storey, and colleagues, "The SPACE of Developer Productivity," ACM Queue, 2021. <https://queue.acm.org/detail.cfm?id=3454124>
24. Marilyn Strathern, "'Improving ratings'. Audit in the British University system," European Review, 1997, p. 308.



Run this with your team.

Hamster is free for 10 Briefs a month, with unlimited viewers.

tryhamster.com/get-started

This guide on the web: tryhamster.com/guides/before-the-agents