LARRIDIN

Building AI-native engineering teams

Some teams are already many times more productive. Most are still stuck at 20 percent. What separates the best from the rest.

SPEAKER

Ameya Kanitkar · Co-founder and CTO, Larridin

THE PROMISE

In the next few minutes…

LARRIDIN

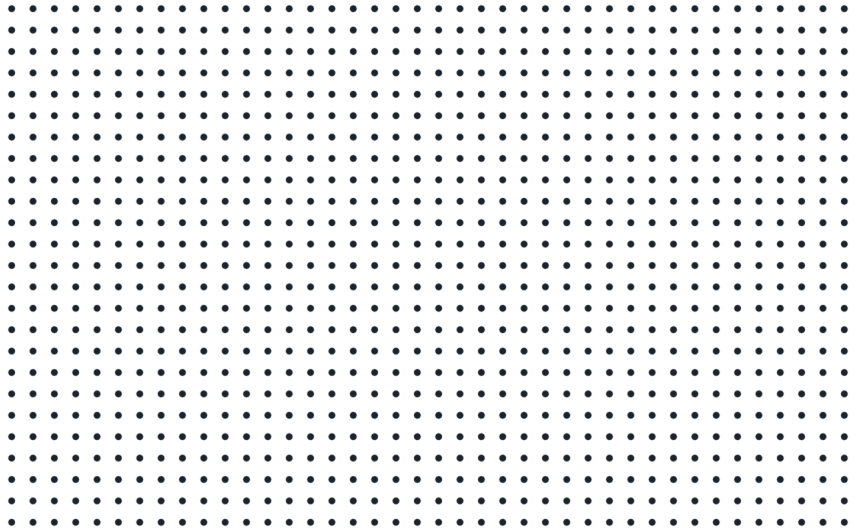
- Going beyond coding tools adoption: Rethink your entire SDLC for AI Native Teams
- What are some common mistakes?
- Right Metrics: Measuring Developer Intelligence in the AI Native Era
- Case Studies
- Changes you can implement TOMORROW

FOR EVERYONE IN THIS ROOM

Up to $7,500 in Larridin Router credits

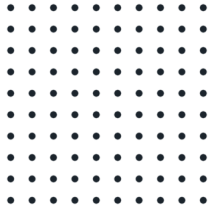
Scan the code or go to larridin.com/leaddev/7500. We follow up to confirm eligibility and activation.

Agents write code 20× faster. Why is productivity up only 20 percent?



A 1,000-PERSON TEAM

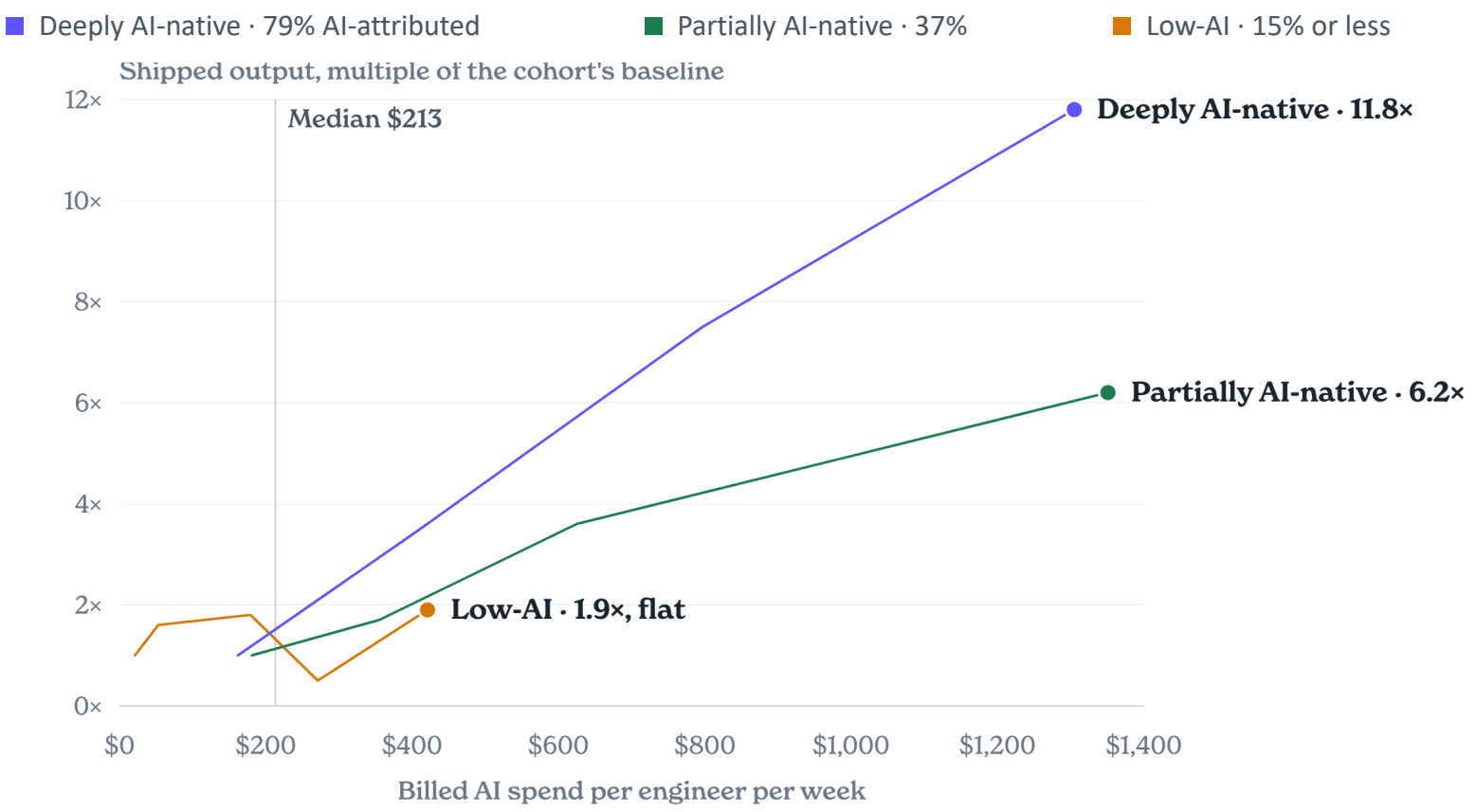
≠



100 PEOPLE

Everything feels faster. But nobody has replaced a 1,000-person team with 100, or a 100-person team with 10.

Some teams are already many times more productive.



$213

Median billed AI spend per engineer per week

10×

The p90 engineer spends more than ten times the p25 engineer

Flat

Low-AI cohort output across a 20× spend range

11.8×

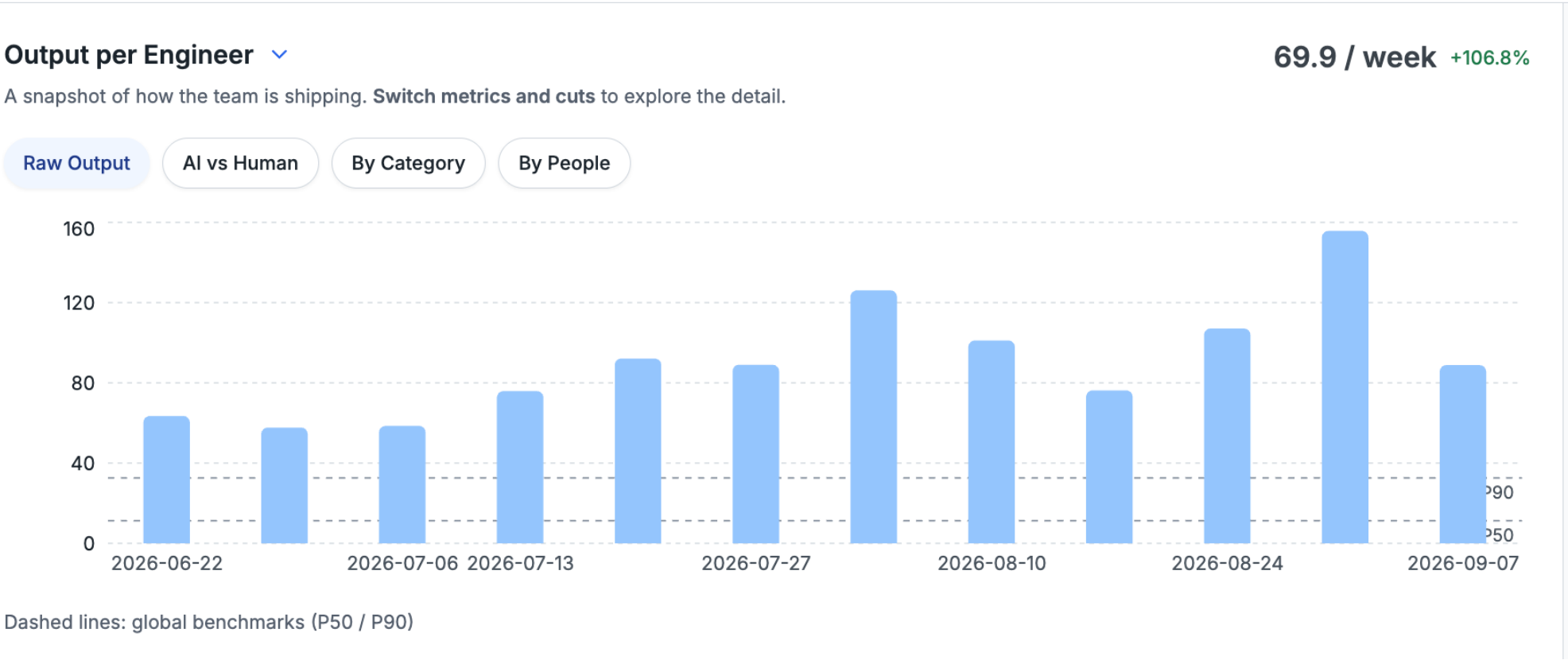
Deeply AI-native cohort, still compounding at about $1,300 a week

Source: Larridin Benchmark, four complete weeks ending Aug 2, 2026. Engineers who both shipped code and drew billed spend, across 100+ software teams.

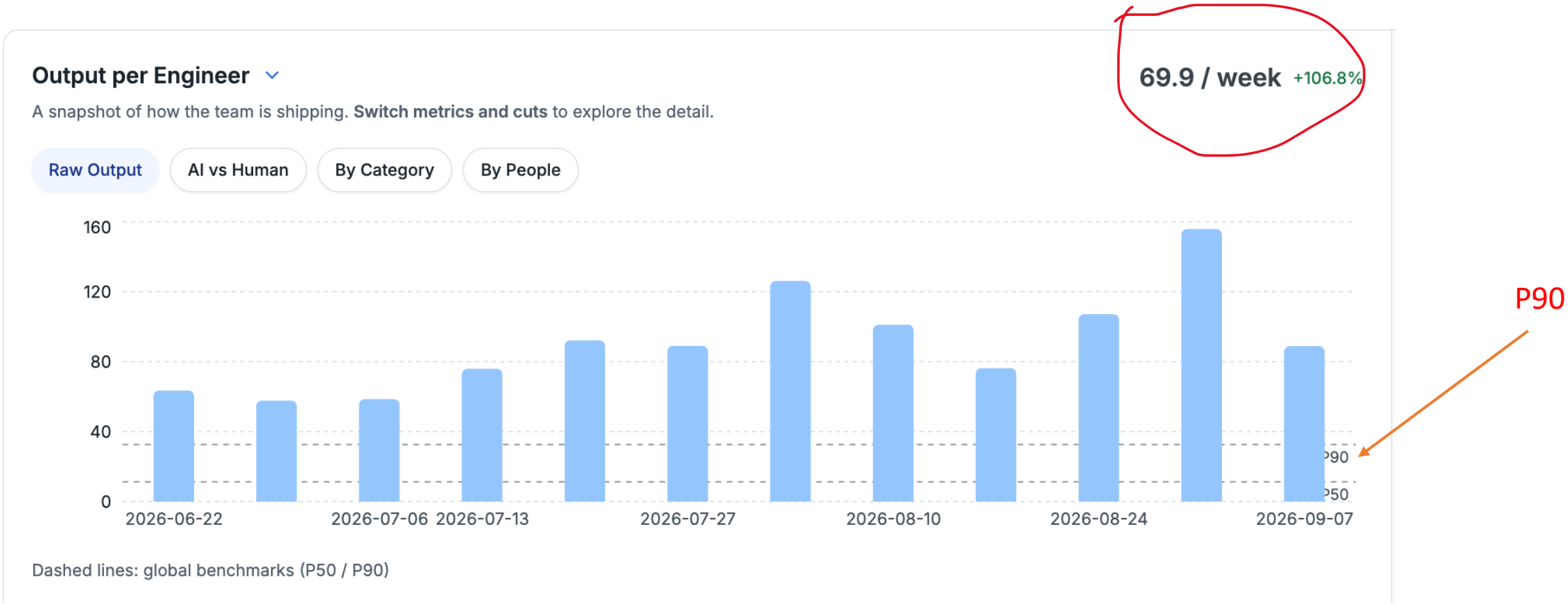
Some teams are already many times more productive.

Source: Larridin Benchmark, four complete weeks ending Aug 2, 2026. Engineers who both shipped code and drew billed spend, across 100+ software teams.

FROM OUR OWN EXPERIENCE



FROM OUR OWN EXPERIENCE



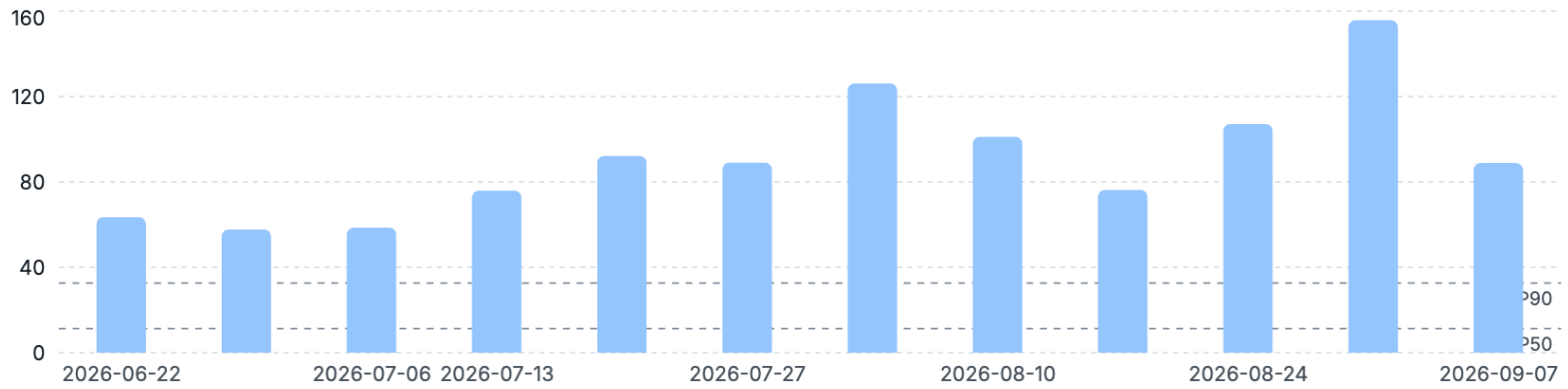
P90

Output per Engineer

A snapshot of how the team is shipping. **Switch metrics and cuts** to explore the detail.

69.9 / week +106.8%

Raw Output | AI vs Human | By Category | By People



Dashed lines: global benchmarks (P50 / P90)

Q3, 2026

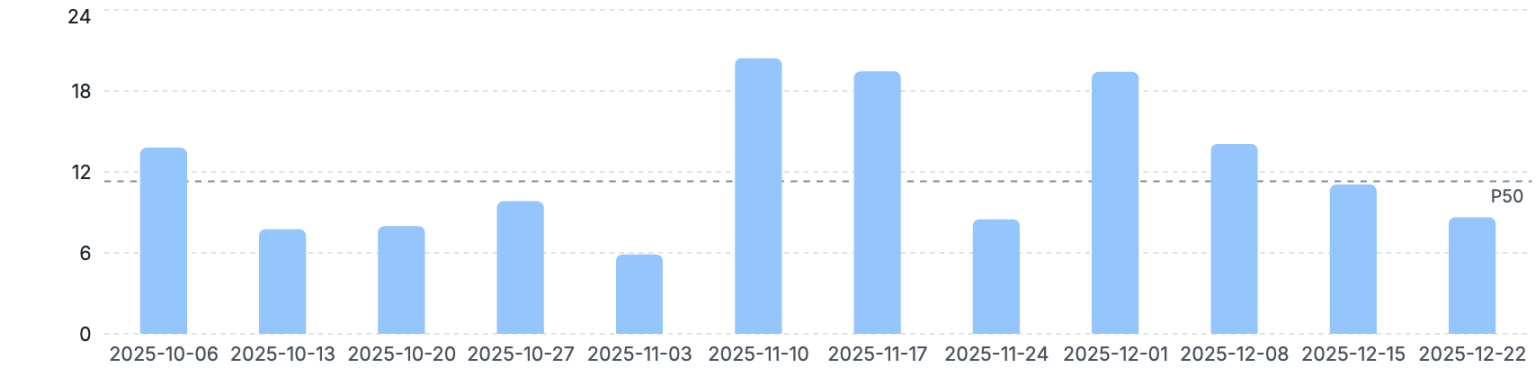
Engineering output up 600%

Output per Engineer

A snapshot of how the team is shipping. **Switch metrics and cuts** to explore the detail.

10.32 / week +2212.58%

Raw Output | AI vs Human | By Category | By People



Dashed lines: global benchmarks (P50 / P90)

Q4, 2025

We are not alone. Many companies large and small achieving these results.

And making strategic structural bets …

Shopify is moving its mobile apps back to native.

In 2020 they went all-in on React Native to share one codebase. In September 2026 they reversed it.

Agents now do enough of the implementation, translation, testing and review that building the same feature twice, in Swift and Kotlin, is cheaper than maintaining one shared codebase.

2020

One shared codebase

React Native. Build once, ship to both platforms, staff from any background.

↓

2026

Two native codebases, agent-maintained

Swift and Kotlin. Agents carry the duplication, translation and parity work.

12 weeks

Shop app, proof of concept to published, in Swift and Kotlin

300+

Screens in the main Shopify app, migration underway

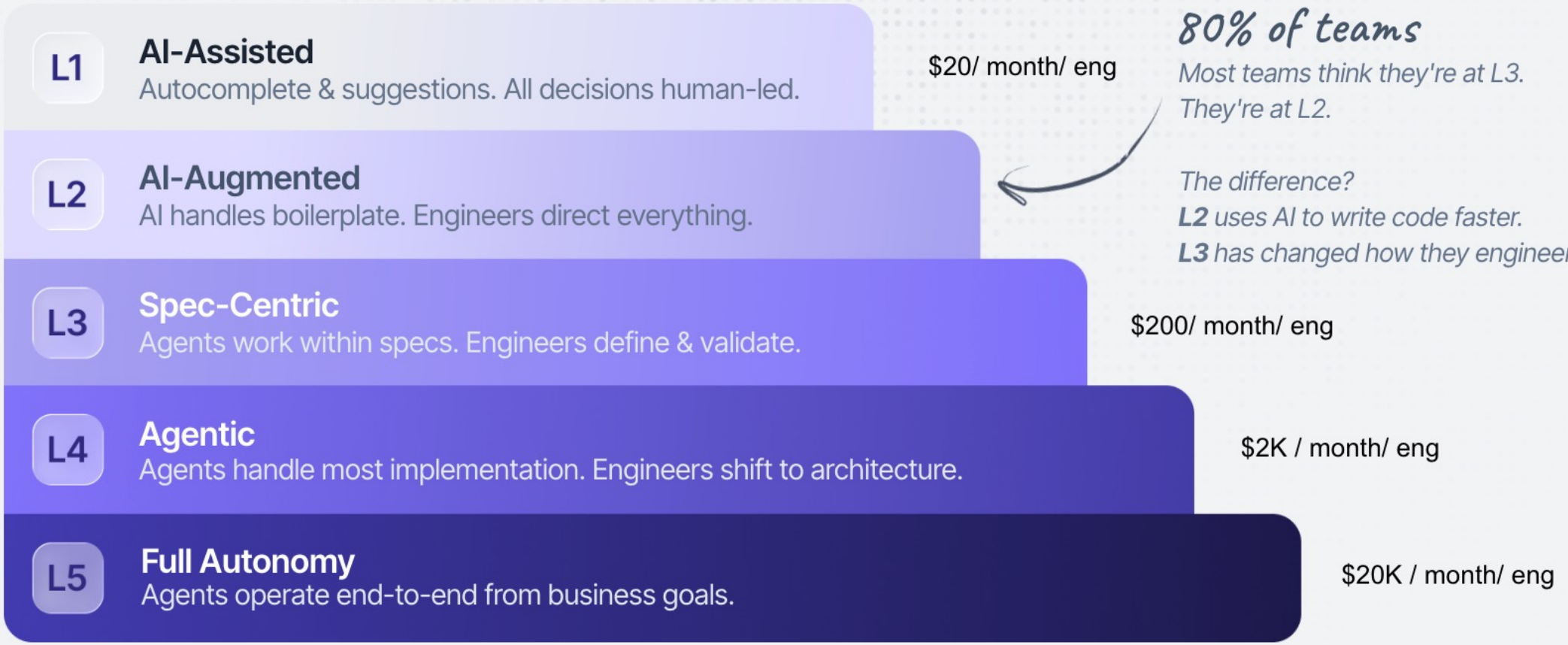
Source: Shopify Engineering, "Native is now the future of mobile at Shopify", September 10, 2026. shopify.engineering/back-to-native

Meta is spending at agentic scale.

~$5B in tokens

with Anthropic, which works out to roughly $15,000 per engineer per month in tokens.

5 Levels of AI-Native Engineering



What separates the best from the rest? Not the tools. The system around them.

THE REST

Adopted the tools. Kept the lifecycle, the review, the environment and the metrics that were built for humans writing every line.

THE BEST

Changed how they engineer. Specs before code, review moved up a level, verification as the precondition, an environment built for agents.

Two sets of changes.

ORGANISATIONAL

Structure, span of control, talent, budgets

How the organisation around the work changes.

DirectorPlus plenary → this afternoon

SYSTEM

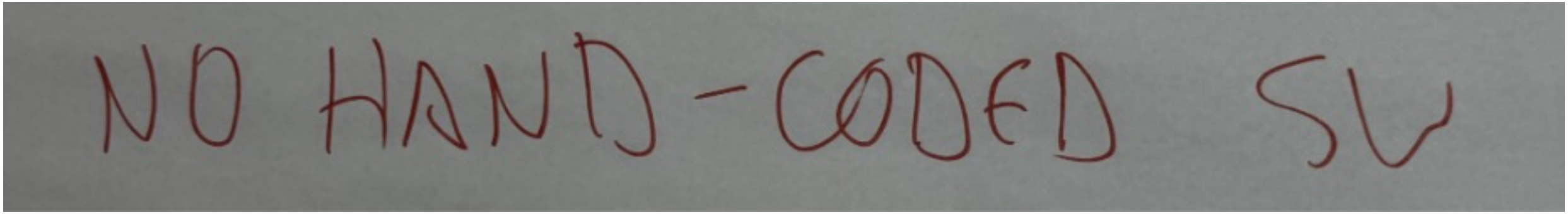
SDLC, verification, environment, measurement

How the system that produces the work changes.

This talk

Step by Step Playbook to 10X Eng Productivity Gains

Rethink the SDLC


NO HAND-CODED SW

Key Changes to SDLC

1. Frontier Intelligence for Planning phase

Use the highest effort setting for the work that matters.

Route the routine work elsewhere.

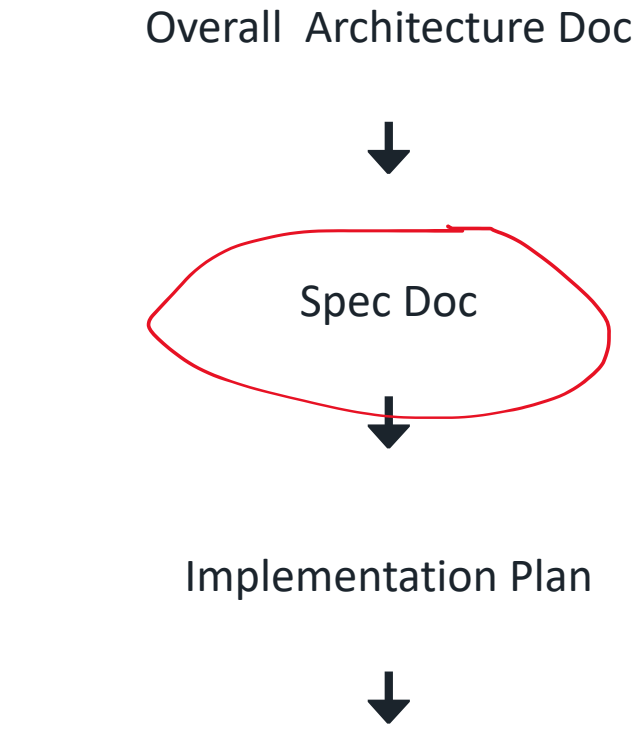
Key Changes to SDLC

2. No code before the spec

Strictly no writing code until the specification is finalized, checked in and reviewed.

Understanding the Spec

KEY TENET: CODE is FREE. Planning and Thinking and Architecture is not



Rule of thumb:
If architecture doc is 200 lines
And actual code is 5000 lines

spec is between 400-600 lines

So, its 10 times small than code, but has the gist of implementation

A few trends I see across the software engineering industry:

- **The death of the IDE.** It's still there, but used less and less. A very sudden change, and IDK about you but I didn't expect it like this.
- **The end of tokenmaxxing.** Tokenmaxxing? What tokenmaxxing??
- **Code reviews in 'zombie' state.** It's mostly theatre reviews now, even at places that say they still do reviews.
- **The boon in using open models for cost savings.** SOTA is great, but also too expensive. Open models + inference much cheaper and nearly as good.
- **The death of middle management.** And with it, career development, to a large extent.
- **Working more than ever.** AI is making is more productive... so why is everyone working so much more??
- **Hiring is... hard?** Finding good engineers is touch. *Unless you're a leading AI lab: then it's hard in different ways: too many incredible people at the door, wanting to be let in, surprisingly strong profiles rejected.*

THE PRAGMATIC ENGINEER

Gergely Orosz
@GergelyOrosz

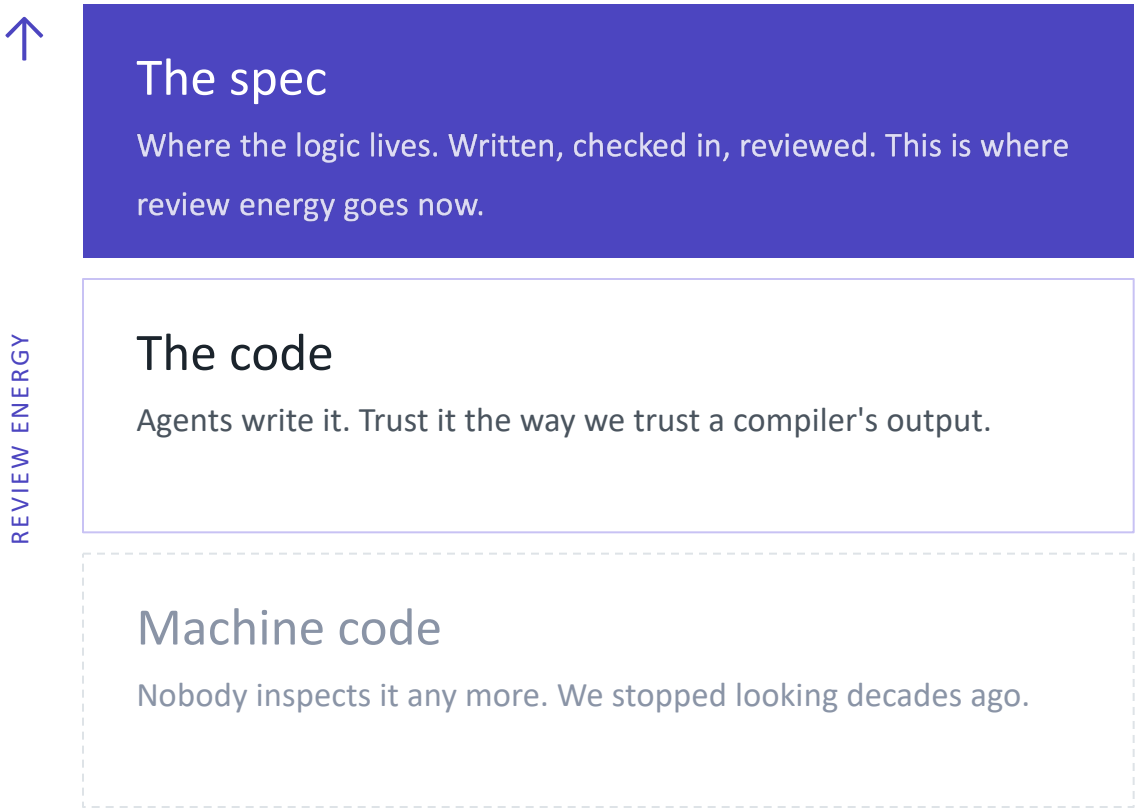
A couple of trends I'm seeing across the industry. ...

So.... no code reviews?????

Move review one level up: from code to spec.

Nobody inspects machine code anymore. We trust the compiler. Treat coding agents the same way. But the logic still has to be verified, so put the review energy where the logic lives.

A management emphasis change, not a tooling purchase



Key Changes to SDLC

3. Don't trust a single agent

Chain models: Codex → Claude → Kimi -> Grok. No model certifies its own work.

Key Changes to SDLC

4. Tests before the code

Test-driven development is not optional when agents write the code.

Verification is the precondition.

If agents produce the volume, the output has to be self-certifiable.

Agents don't fail randomly. They fail when context is lost.

That tells you exactly where to concentrate verification effort.

TDD becomes non-negotiable

Tests are the constraints that make agent output trustworthy.

Tests stay human-readable

Even when the code no longer is.

No model checks its own work

Never trust a single model to certify what it produced.

MUST DO SDLC Changes

01

Frontier Intelligence for Planning phase

Use the highest effort setting for the work that matters. Route the routine work elsewhere.

02

No code before the spec

Strictly no writing code until the specification is finalised, checked in and reviewed.

03

Don't trust a single agent

Chain models: Codex → Claude → Kimi. No model certifies its own work.

04

Tests before the code

Test-driven development is not optional when agents write the code.

If you follow one piece of advice: install Superpowers.

obra / superpowers

Type / to search

ode Issues 140 Pull requests 142 Actions Projects Security and quality Insights

superpowers Public

Sponsor Watch 795 Fork 18.5k Starred 208k

main 72 Branches 27 Tags Go to file T Add file Code

About

An agentic skills framework & software development methodology that works.

Readme

11 people Release v5.1.0 (#1468) f2cbfbe · 3 weeks ago 440 Commits

.claude-plugin Release v5.1.0 (#1468) 3 weeks ago

Superpowers: How Jesse Built the #1 AI Claude Code/ Codex Plugin — and Stopped Writing Code

Build for agents first

Boris Cherny
@bcherny

𝕏.com

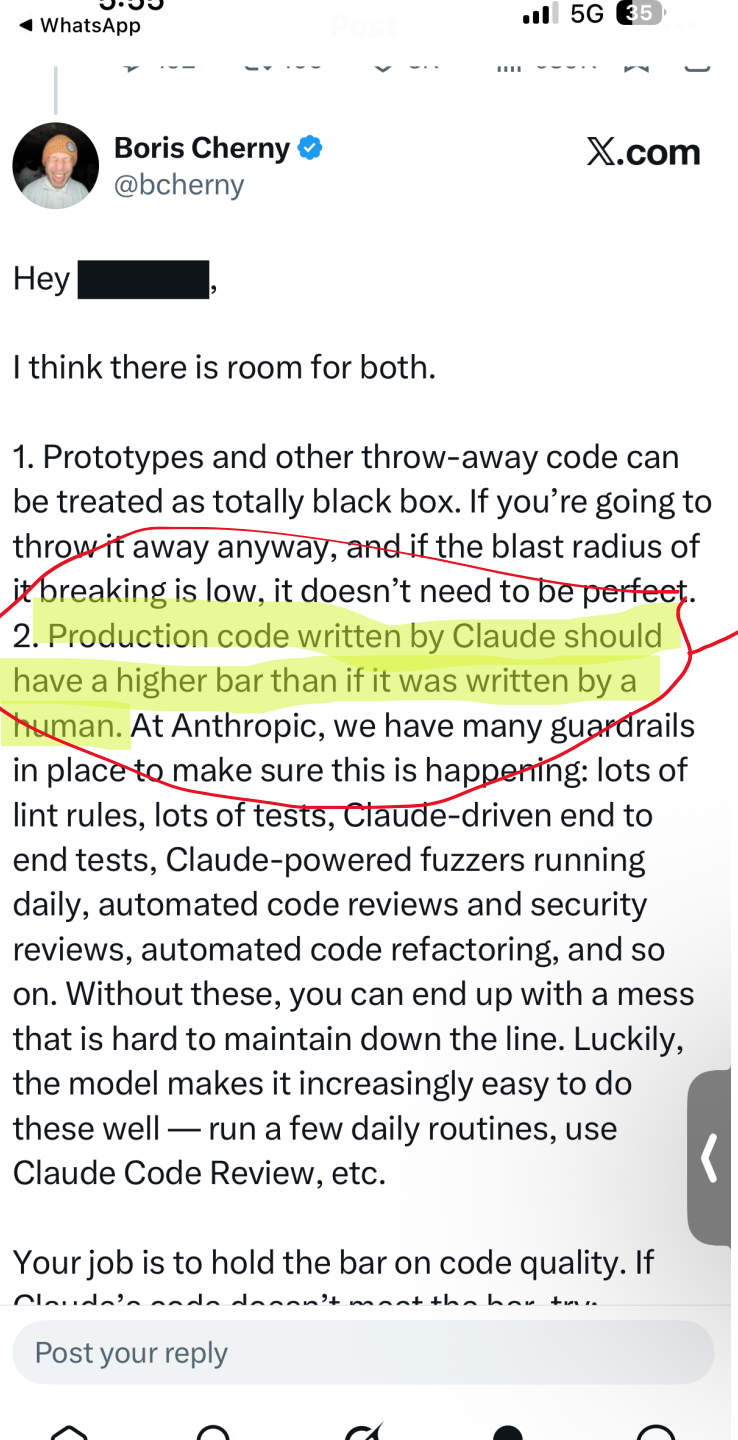
Hey [redacted],

I think there is room for both.

1. Prototypes and other throw-away code can be treated as totally black box. If you're going to throw it away anyway, and if the blast radius of it breaking is low, it doesn't need to be perfect.
2. Production code written by Claude should have a higher bar than if it was written by a human. At Anthropic, we have many guardrails in place to make sure this is happening: lots of lint rules, lots of tests, Claude-driven end to end tests, Claude-powered fuzzers running daily, automated code reviews and security reviews, automated code refactoring, and so on. Without these, you can end up with a mess that is hard to maintain down the line. Luckily, the model makes it increasingly easy to do these well — run a few daily routines, use Claude Code Review, etc.

Your job is to hold the bar on code quality. If Claude's code doesn't meet the bar, try:

Post your reply



Production code written by an agent should have higher bar than written by a human

Your environment has a new primary user.

CI pipelines	Version control	Codebases	Documentation
BUILT FOR HUMANS	BUILT FOR HUMANS	BUILT FOR HUMANS	WRITTEN FOR HUMANS TO READ

The primary user changed. The environment didn't.

Build for agents first.

We built everything for humans. Now the environment has to be easy for agents to navigate, verify against, and operate in, safely and autonomously.

01

AGENTS.md quality

Context maintained like code, not wiki rot.

02

Codebase structure

The monorepo question is now an agent-navigation question.

03

Machine-readable context

Docs, runbooks and interfaces agents can consume.

04

Safe autonomous execution

Paths where agents can act without a human in the loop.

05

Agent tech debt

A real category, with a real owner.

Build for agents first.

We built everything for humans. Now the environment has to be easy for agents to navigate, verify against, and operate in, safely and autonomously.

01

AGENTS.md quality

Context maintained like code, not wiki rot.

02

Codebase structure

The monorepo question is now an agent-navigation question.

03

Machine-readable context

Docs, runbooks and interfaces agents can consume.

04

Safe autonomous execution

Paths where agents can act without a human in the loop.

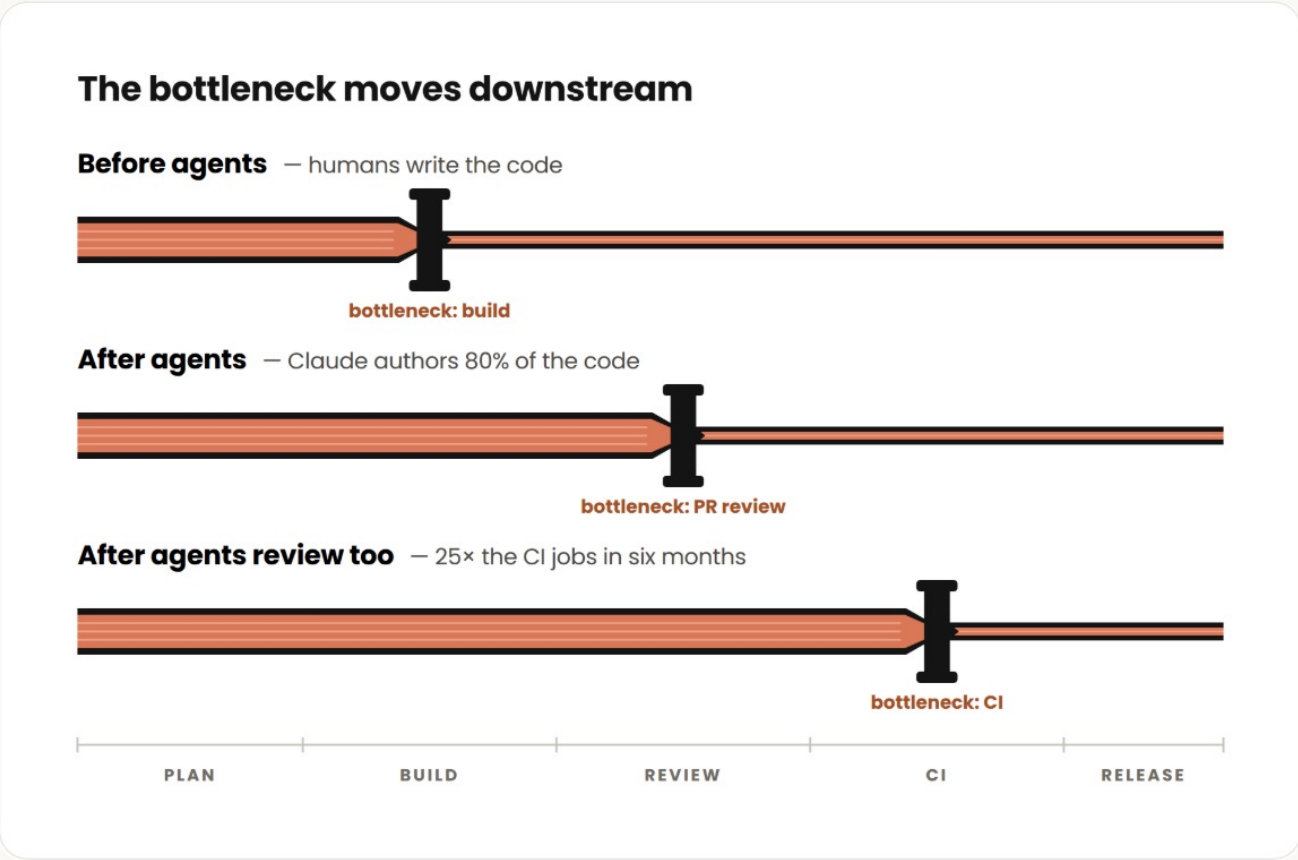
05

Agent tech debt

A real category, with a real owner.

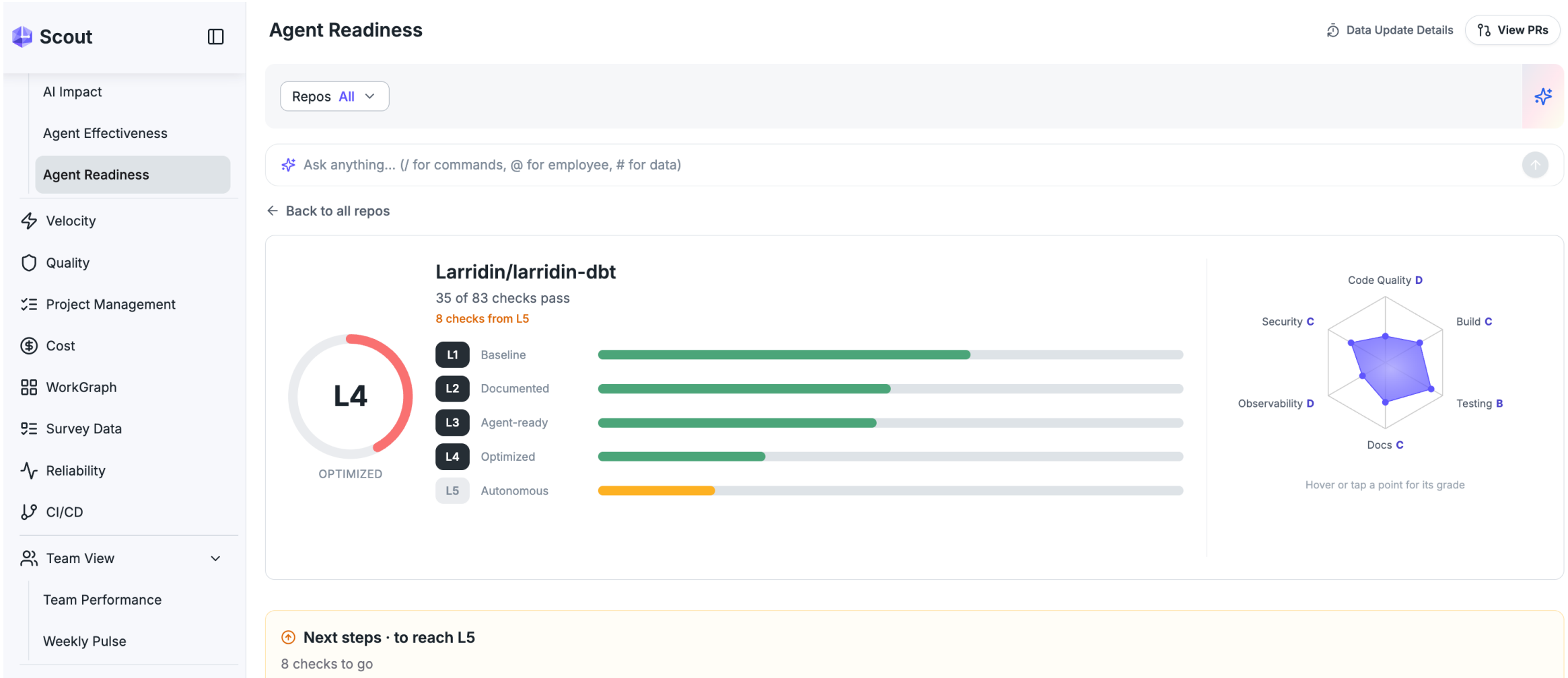
Build for Agents

Anthropic: 25x increase in CI jobs in 6 Months

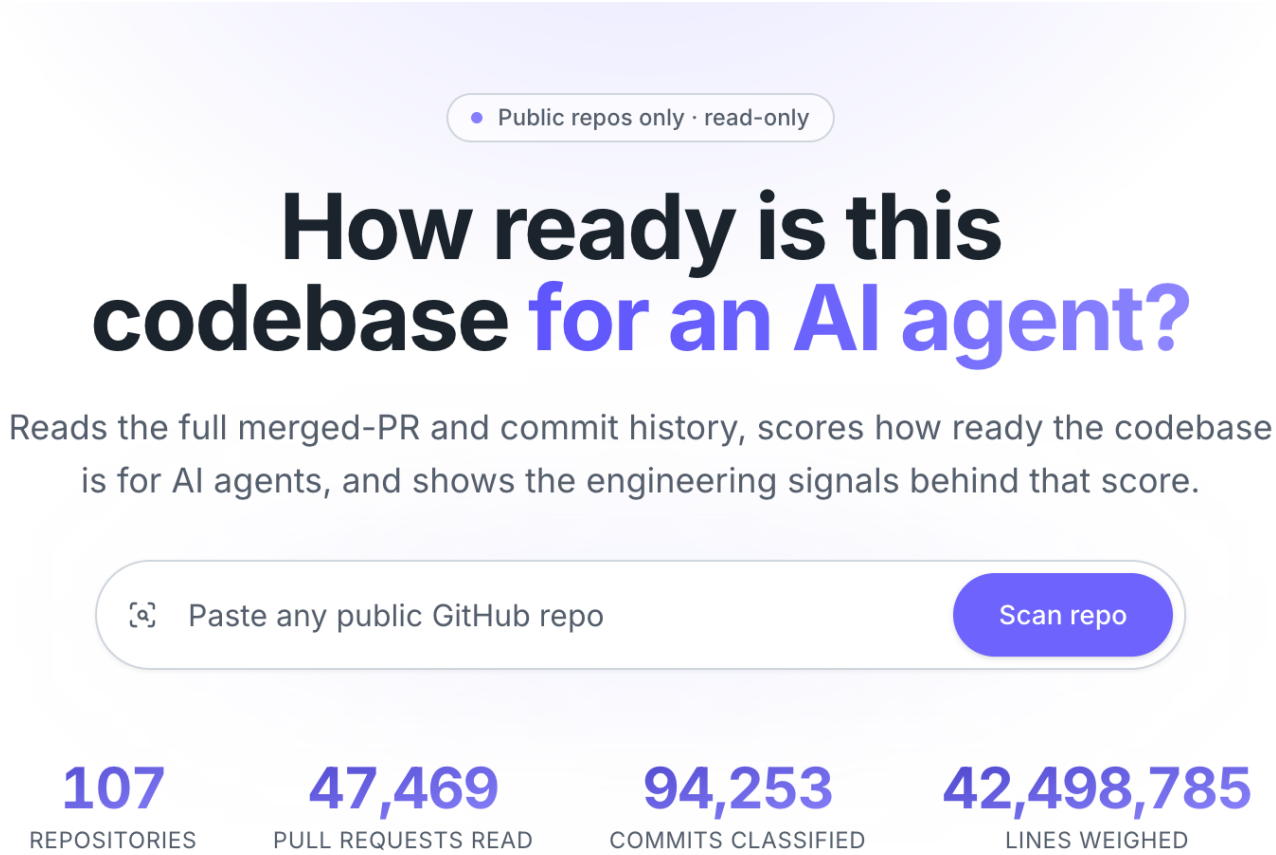


Writing code is no longer the constraint, and once PR review gets accelerated, CI starts feeling the pressure.

Build for agents first.

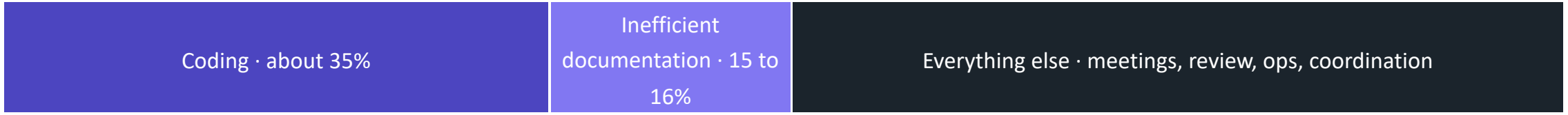


Free readiness assessment:
http://code.larridin.com



How you know it's working

Coding was never the bottleneck.



Make the 35% infinitely fast. You still cap out.

Only about a third of engineering time was ever spent writing code. Inefficient documentation alone can account for 15 to 16 percent of total engineering time.

The Work Graph: where the time actually goes.

Only 30 to 40 percent of engineering effort is coding.

This is visibility into the rest.

EFFORT SPLIT · BOX AREA = SHARE OF ENGINEERING EFFORT

Live product view, all teams, last 12 weeks, June 1 to August 23, 2026. Each tile opens the intent breakdown.

The metrics framework.

Adoption

Are teams using AI tools? Active users, engagement with AI tools.

Velocity

Complexity-adjusted throughput. Output adjusted for quality.

Quality

Code Durability, rework rate, escaped defects.

Cost and ROI

Token economics. Spend against shipped output.

PLUS Agent telemetry: how humans actually work with agents. Genuinely new signal.

The first 90 days.

"Rethink your SDLC" is unactionable without sequencing.

START MONDAY

Pick one team: specs written, checked in, reviewed.

TDD required for agent-written code.

Baseline the mechanism: spend per engineer, and where the time goes.

BY DAY 90

Shift review energy from code to spec across the pilot teams.

Independent verification gates. No model certifies its own work.

Agent-readiness audit of the top repos. Name the owner for agent tech debt.

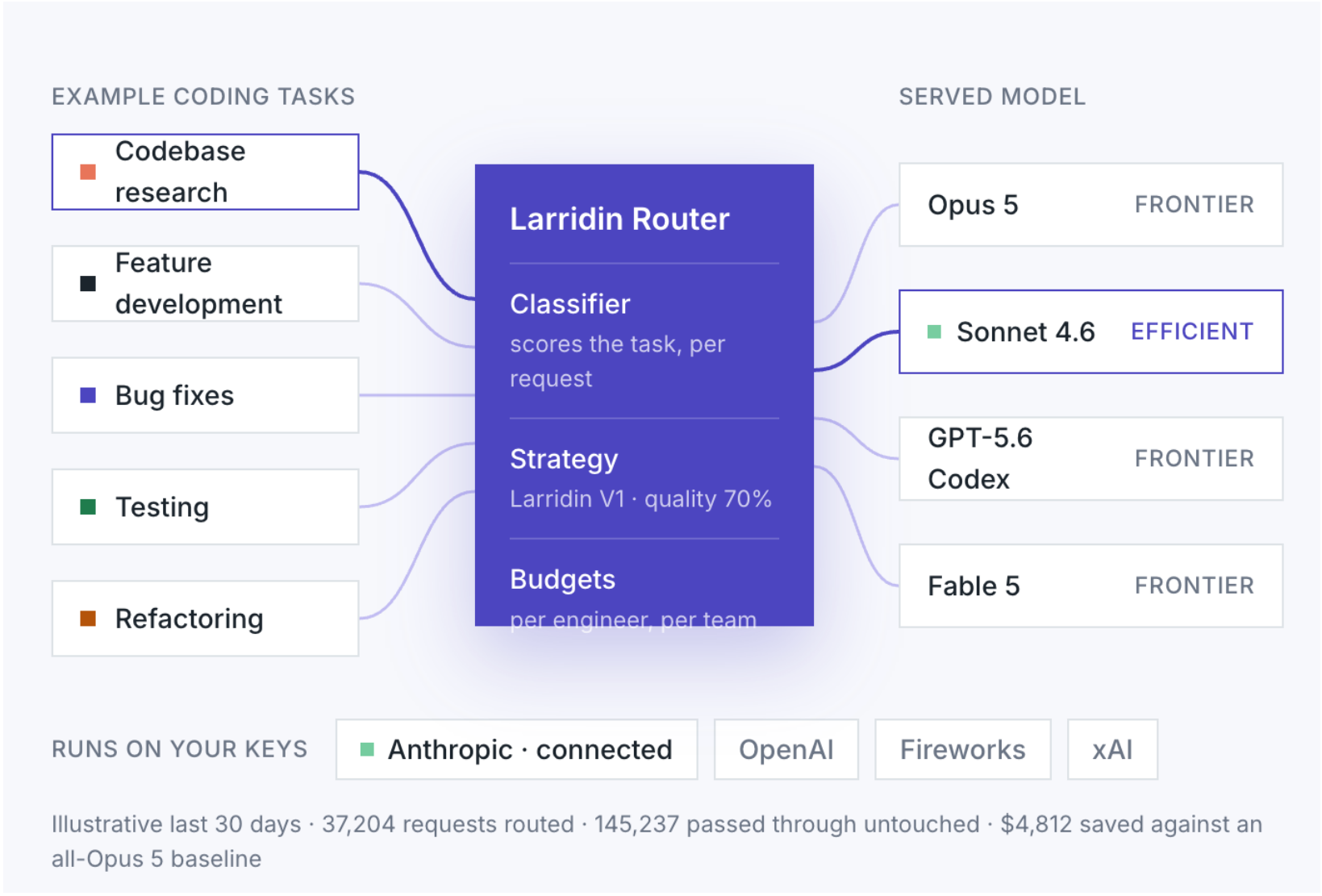
HONESTLY, A YEAR

The environment rebuilt agent-first.

Budgets tied to AI-nativeness, decided on data.

The organisational half: structure, talent, span of control. This afternoon's plenary.

Get a Router!


EXAMPLE CODING TASKS
Codebase research
Feature development
Bug fixes
Testing
Refactoring
Larridin Router
Classifier
scores the task, per request
Strategy
Larridin V1 · quality 70%
Budgets
per engineer, per team
SERVED MODEL
Opus 5
FRONTIER
Sonnet 4.6
EFFICIENT
GPT-5.6 Codex
FRONTIER
Fable 5
FRONTIER
RUNS ON YOUR KEYS
Anthropic · connected
OpenAI
Fireworks
xAI
Illustrative last 30 days · 37,204 requests routed · 145,237 passed through untouched · $4,812 saved against an all-Opus 5 baseline

Thank you.

The organisational half is this afternoon: the DirectorPlus plenary on structure, talent and budgets.

Up to $7,500 in Larridin Router credits for everyone in this room. We follow up to confirm eligibility and activation.

larridin.com/leaddev/7500

Scan for your credits

larridin.com/leaddev/7500

CONTACT

Ameya Kanitkar · ameya@larridin.com