encore

The CTO's Guide to Post-AI Engineering Economics

Where the gains go, what the bottleneck costs, and when to buy instead of hire.

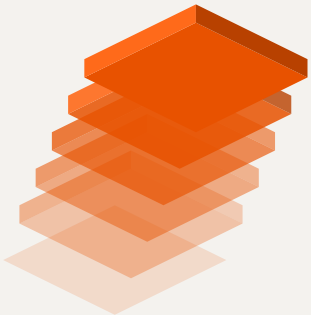


Table of Contents

Executive Summary

Two out of three software firms have rolled out generative AI tools. Bain puts the typical productivity gain at 10 to 15%, and notes the time saved is often not redirected toward higher-value work. Companies that pair the tools with end-to-end process transformation report 25 to 30%.[1] McKinsey's economy-wide survey finds only a small minority of companies reporting significant bottom-line impact from AI.[2] The tools deliver measurable individual gains, and in most organizations the company-level economics have not caught up yet.

This guide is about that difference, in the language of a budget owner: where the gains disappear between the individual developer and the P&L, what the delivery bottleneck costs in money, how AI changes the case for the marginal engineering hire, when to build a platform team versus buy a platform, and how to put the whole argument in front of a board.

[1] Bain & Company, "From Pilots to Payoff: Generative AI in Software Development", Technology Report 2025. bain.com

[2] McKinsey, "The State of AI in 2025: Agents, Innovation, and Transformation", Nov 2025, n=1,993. 109 of 1,993 respondents (5.5%) meet the "AI high performer" bar; 39% report any EBIT impact.

1 Where the AI Gains Go

Between adoption and the income statement, the AI investment loses value at every step: adoption is nearly universal, individual gains are real but modest, and organizational impact remains rare.

FIGURE 01 — WHERE THE AI GAINS STOP

ADOPTION → P&L

80%+
OF COMPANIES USE GENERATIVE AI FOR CODING[3]

10–15%
TYPICAL PRODUCTIVITY GAIN FROM AI CODING TOOLS ALONE[1]

~5%
QUALIFY AS MCKINSEY "AI HIGH PERFORMERS": OVER 5% OF EBIT ATTRIBUTABLE TO AI[2]

Adoption is near universal, the individual gain is modest, and company-level impact stays rare. Sources: BCG, Bain and McKinsey, as footnoted on this page.

The shortfall sits downstream of the tools. Writing and testing code is only 25 to 35% of the time from idea to launch, so even a large gain on that slice moves the whole pipeline modestly.[1] BCG's verdict on engineering organizations is blunt: widespread adoption, shallow impact, with value "spotty and limited to pockets", and half of CIOs unable to quantify GenAI's impact at all.[3]

Where the gains go

- → **Review queues.** AI-generated pull requests wait 4.6x longer for first review; acceptance runs 32.7% versus 84.4% for manual PRs.[4]
- → **Organizational drag.** 68% of developers save 10+ hours a week with AI; 50% lose 10+ hours a week to inefficiencies like waiting on environments and hunting for information.[5]
- → **Instability.** AI adoption continues to correlate with degraded delivery stability, and incident response consumes the saved time faster than the tools created it.[6]
- → **Unmeasured impact.** 90% of engineering teams use AI tools; only 20% measure the impact with engineering metrics, so most teams have no way to see where the saved time goes.[7]

THE PROCESS-TRANSFORMATION GAP

Bain's gap between 10-15% and 25-30% amounts to roughly a doubling of AI return, and closing it depends on rebuilding the delivery process around the tools you already have rather than buying better ones.[1]

3 BCG, "Executive Perspectives: AI-Enabled Engineering Excellence Transformation", Apr 2025.

4 LinearB, "2026 Software Engineering Benchmarks Report", 8.1M+ pull requests, 4,800+ organizations.

5 Atlassian, "State of Developer Experience 2025", 3,500 developers and managers, with Wakefield Research.

6 Google Cloud DORA, "State of AI-assisted Software Development", 2025.

7 Jellyfish, "2025 State of Engineering Management Report", Jul 2025.

2 How AI Changes the Engineering Budget

AI has rearranged what engineering money buys rather than making engineering cheaper. Four line items are moving at once:

FIGURE 02 — FOUR BUDGET LINES MOVING AT ONCE

ENGINEERING P&L

LINE ITEM	WHAT IS CHANGING
AI tooling and models	A new, growing budget line. AI-enabled companies now allocate 10 to 20% of R&D budgets to AI, and the share is growing across every revenue band.[8]
Headcount composition	Organizations expect 20 to 30% of engineering headcount to be AI-focused. The evidence so far shows the mix of roles changing while totals hold: AI is not yet driving significant headcount reductions.[8]
Hiring posture	The sector runs leaner: US tech job postings are down roughly a third from their 2020 baseline, while engineering has been among the most resilient functions.[9] Teams are expected to do more per head.
Delivery overhead, usually unbudgeted	Review queues, environment waits, infrastructure work, and incident response. This is where the stalled gains from chapter 1 land, and in most budgets it is not itemized at all.

Where AI has rearranged engineering spend. Sources: ICONIQ 2025 for tooling and headcount composition; Indeed Hiring Lab and SignalFire for hiring posture, as footnoted on this page.

The measurement gap

90% of engineering teams use AI tools. 20% measure the impact with engineering metrics.[7] For a budget owner, that means most organizations are increasing AI spend without being able to say which part of it returns anything, and without seeing that the constraint has moved downstream of the tools. The first economic decision is instrumentation: attribute AI usage, track the delivery metrics, and put a number on the delivery overhead line.

AI AND ENGINEERING HEADCOUNT

No credible dataset yet shows AI reducing engineering headcount at scale. What the data shows is leaner hiring, more roles pointed at AI work, and more expected from each engineer.[8,9] Plan for more output per engineer, with payroll roughly flat.

8 ICONIQ, "The State of AI 2025: The Builder's Playbook", ~300 software executives surveyed, Jun 2025.

9 Indeed Hiring Lab, "The US Tech Hiring Freeze Continues", Jul 2025 (US tech postings 36% below Feb 2020); SignalFire, "State of Tech Talent Report 2026" (Tech Major hiring -25% vs 2019, engineering -11%).

3 What the Bottleneck Costs

Three numbers put a price on the delivery bottleneck, and all three belong in any board conversation about engineering velocity:

FIGURE 03 — WHAT THE BOTTLENECK COSTS

DOWNTIME, 2026

$600B
ANNUAL COST OF UNPLANNED DOWNTIME FOR GLOBAL 2000 COMPANIES[10]

$15k
AVERAGE COST PER MINUTE OF DOWNTIME[10]

$95M
AVERAGE ANNUAL LOST REVENUE PER ORGANIZATION FROM DOWNTIME[10]

The three downtime numbers that belong in a board conversation about velocity. Source: Splunk (Cisco) with Oxford Economics, 2026, as footnoted on this page.

Downtime is the most visible of these costs, and AI is currently adding to it: adoption keeps correlating with degraded delivery stability, and 72% of organizations have had at least one production incident caused by AI-generated code.[6,11] Every point of stability given up converts saved developer hours into incident hours, the most expensive category of engineering time.

Maintenance and queue costs

The older, larger number is maintenance drag. Stripe's canonical study put developer time on maintenance and bad code at 17.3 hours of a 41.1-hour week, roughly 42%, and valued the resulting inefficiency at about $300B a year in global GDP loss. The study is from 2018, before AI raised code volume; nothing suggests the share has fallen since.[12] Add the modern queue costs from chapter 1: payroll spent waiting for review, environments, and redeploy cycles.

WHAT DELIVERY AUTOMATION IS WORTH

Harness's 2025 research found organizations that move from low to moderate continuous-delivery automation more than double their likelihood of realizing velocity gains from AI, from 26% to 57%.[11] In economic terms: at low automation, roughly three quarters of the AI investment's intended return never materializes. How much of the AI budget returns anything depends on the delivery platform underneath it.

The decisions in the next two chapters, whether to add headcount and whether to build or buy the platform layer, are both about the cheapest way to shrink this bottleneck.

10 Splunk (Cisco) with Oxford Economics, "The Hidden Costs of Downtime 2026: A $600 Billion Wake-Up Call", May 2026.

11 Harness, "The State of AI in Software Engineering 2025", Coleman Parkes survey of 900 engineers and leaders, fielded Aug 2025.

12 Stripe with Harris Poll, "The Developer Coefficient", Sep 2018.

4 Headcount or Automation?

The traditional response to a delivery bottleneck was to hire platform and DevOps engineers until the queue cleared. Three things have changed that maths.

Platform hires cost more and take longer to find

In the Kubernetes job market, platform engineers command nearly 20% more than DevOps roles, and platform and IaC roles rank among the hardest to fill.[13] Meanwhile the function's own economics are under scrutiny: 47.4% of platform initiatives run on budgets under $1M, 29.6% of platform teams do not measure their own success at all, and average platform salaries fell in North America and Europe against 2024 as the title went mainstream.[14] Senior platform skills stay scarce while the routine work commoditizes, which is usually the point at which the routine part gets automated.

The work is changing

AI has made application code abundant and cheap without doing the same for the infrastructure work around it, which is why the bottleneck sits there. Hiring more people to shepherd a growing volume of AI-generated change means scaling up the expensive input to keep pace with the cheap one. The alternative is to automate the work away: let provisioning, environments, and deployment derive from the application code, so more code stops meaning proportionally more infrastructure work.

AI AND THE DEVOPS ROLE

The data does not show AI eliminating DevOps roles, and it may never do so in the aggregate: ICONIQ finds workforce composition changing rather than shrinking.[8] What is changing is the content of the role, away from ticket-driven provisioning and pipeline maintenance and toward policy, guardrails, and platform decisions. The useful question for the next requisition is whether it hires someone to do work a platform should be doing.

What to calculate before you sign the requisition

- → Fully loaded cost of the platform hire, times the team you will eventually need. Platform teams rarely stay at one.
- → How much of that capacity goes to undifferentiated plumbing: IaC upkeep, environment drift, CI pipelines.
- → What the same money buys if you spend it removing the work instead: a platform that derives infrastructure from code, or automation of whatever consumes the most time today.
- → How fast each option shows results: a hire lands in months and takes longer to ramp, while a platform change on one service shows a difference in weeks.

13 Kube Careers, "State of the Kubernetes Job Market Q1 2025" (436 Kubernetes-required postings); Robert Half, 2026 Salary Guide.

14 Platform Engineering Community, "State of Platform Engineering Report Vol. 4", Dec 2025, n=518 (fielded Aug-Oct 2025).

5 Build or Buy: Platform Engineering in the AI Era

Gartner predicted that by 2026, 80% of large software engineering organizations would run platform engineering teams, up from 45% in 2022.[15] The prediction is arriving on schedule, but it says nothing about whether those platforms should be built internally or bought, and the AI era has moved that answer.

FIGURE 04 — BUILD OR BUY, LINE BY LINE — PLATFORM LAYER

	BUILD INTERNALLY	BUY A PLATFORM
Cost shape	A standing team, growing with scope. Nearly half of platform initiatives run under $1M/yr, which buys 3-5 engineers and little else[14]	Subscription plus adoption effort. Cost scales with usage
Time to value	Quarters to years before developers feel it	First service in days; delta measurable in weeks
Risk profile	Internal platforms compete with product for your best engineers; 29.6% of platform teams cannot demonstrate their own value.[14] Gartner expects over 40% of agentic AI projects to be canceled by end of 2027, in part on inadequate risk controls and unclear value[16]	Vendor risk: how locked in you become, and how much you depend on someone else's roadmap. Open frameworks and deployment into your own cloud account limit both
AI readiness	Building agent-safe guardrails and code-derived provisioning in-house is a product-scale effort	Arrives as the platform's core competency, maintained against the moving AI landscape for you

The same platform layer priced both ways. Sources: Platform Engineering Community, Vol. 4, and Gartner, as footnoted on this page.

Whether to build comes down to differentiation: if your platform layer is a competitive asset, build it. For most companies it is undifferentiated plumbing that has to work well and wins nothing competitively, and DORA's finding sets the stakes for getting it right by either path: a high-quality platform amplifies the effect of AI adoption on organizational performance, and a mediocre one caps it.[6]

A TEST FOR BUILD VERSUS BUY

Would you staff this platform with your five best engineers for the next three years, and would your board approve that allocation if it appeared as its own budget line? A no to either usually decides the debate in favor of buying.

15 Gartner, platform engineering trend prediction, 2023, reiterated on gartner.com through 2026.

16 Gartner, "Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027", press release, Jun 25, 2025 ("escalating costs, unclear business value or inadequate risk controls").

6 The Economics of Code-Derived Infrastructure

The buy option worth pricing is infrastructure derived from application code: developers declare what a service needs as typed code, and the platform provisions, deploys, and observes it. On the P&L, the model changes five lines:

FIGURE 05 — FIVE P&L LINES

TRADITIONAL VS DERIVED

P&L LINE	TRADITIONAL STACK	CODE-DERIVED INFRASTRUCTURE
Infrastructure headcount	Grows with service count and change volume	Provisioning, environments, and deploys are automated; hires go to differentiated work
Time to market	The 65-75% of idea-to-launch time that is not coding sets the pace[1]	Customer-reported: 2-3x development speed, 90% shorter lead times[17]
Incident exposure	Drift, config errors, and AI-speed change against manual controls; $15k per downtime minute[10]	Identical environments and typed guardrails remove the config-error class
Review capacity	Human review scales linearly with AI output	Platform gates absorb infrastructure review; humans review business logic
Return on AI tools	10-15% and leaking[1]	The 25-30% Bain attributes to companies that rebuild the process: AI works in app code, delivery is automatic[1]

How code-derived infrastructure changes each line. Sources: Bain 2025, Splunk 2026, and Encore customer case studies, as footnoted.

> "Encore lets us scale both our product and infrastructure footprint, without additional hiring. The ROI we've seen is outstanding, easily 10x."
>
> **Daniel Stocks, CTO**
>
> CARLA · THE CASE STUDY REPORTS ANNUAL SAVINGS OF OVER $200,000 ON HEADCOUNT COSTS

Customer-reported numbers are outcomes from specific contexts rather than guarantees.[17] Use them as hypotheses for your own pilot, measured on your own baseline; chapter 8 covers the setup.

[17] Encore customer case studies (Carla; Groupon). Customer-reported outcomes. encore.dev/customers

7 Making the Case to the Board

A board judges engineering investment by revenue timing, cost exposure, and risk, so DORA metrics need translating before they reach the deck. Once you have the measurements from chapter 2, the translation is direct:

FIGURE 06 — ENGINEERING METRICS IN BOARD LANGUAGE TRANSLATION

ENGINEERING METRIC	BOARD LANGUAGE
Lead time for changes	Time from decision to revenue, and what a week of delay costs on the roadmap's top items
Deployment frequency	How often the company can respond to the market: optionality, in board terms
Change failure rate	Incident exposure in dollars: probability times $15k/minute times mean duration[10]
Review-queue and wait time	Payroll spent waiting, and the FTEs you could recover without hiring
% of engineering time on non-feature work	Share of R&D budget not building product. The line AI was supposed to shrink, and can't while infrastructure is manual

Each delivery metric restated as the board hears it. Downtime cost from Splunk with Oxford Economics, footnoted in chapter 3.

The one-page memo

- → **Situation:** AI tooling is deployed; individual gains are real. Delivery metrics are flat, and here are ours.
- → **Diagnosis:** the constraint has moved downstream of the code, into review, environments, provisioning, and deployment. Industry data and our own numbers agree.
- → **Options, with costs:** hire (salary, ramp time, share of the role spent on plumbing), build a platform (team size, timeline, risk), buy a platform (subscription, and what the pilot measured).
- → **Ask:** one new service on a pilot, measured on lead time, deploy frequency, and change failure rate against today's baseline, with a decision in one quarter.

Questions to ask any platform vendor, including us

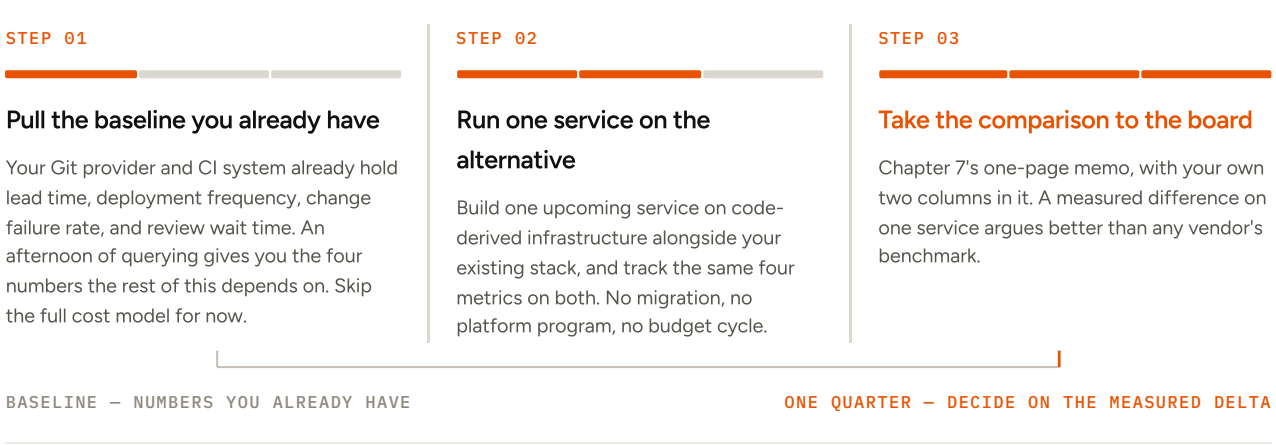
- → What happens to our code and infrastructure if we leave? (Open-source framework, standard cloud resources in our own account, no proprietary runtime.)
- → What does the pilot cost, and what do we measure to judge it? (Near-zero, and the four DORA metrics.)
- → Where does it not fit? (Credible vendors name their exceptions: unusual runtimes, highly specialized infrastructure.)

8 Getting Started

Most engineering organizations cannot answer the questions in this guide today. Only 20% measure AI's impact with engineering metrics at all, so waiting until the full cost picture exists means waiting indefinitely.[7] You do not need it. A decision needs a baseline on three or four metrics you can already pull, and one service to compare against.

FIGURE 07 — THREE STEPS, ONE QUARTER

ONE SERVICE, MEASURED



One upcoming service carries the pilot, alongside the existing stack. Illustrative, not to scale.

The questions the decision eventually turns on

Answer what you can. The gaps are not a prerequisite; several of them close on their own once the pilot is running, and the rest tell you where your reporting is thin.

- → What share of engineering time goes to non-feature work: infrastructure, environments, pipelines, waiting?
- → What is your lead time from merge to production, and how has it moved since AI adoption?
- → How long does an AI-assisted PR wait for review, compared with a human one?
- → What does an hour of downtime cost the business?
- → What did AI tooling cost last quarter, and what measured outcome did it buy?
- → What does the platform and DevOps function cost annually, fully loaded, and how much of that goes to undifferentiated plumbing?
- → If delivery lead time halved, what would that be worth in revenue timing?

DO NOT LET INSTRUMENTATION BECOME THE PROJECT

Building a complete engineering cost model before making any decision is how this stalls for a quarter. The four delivery metrics in step 1 are enough to run a pilot and read the result, and the pilot itself produces most of the numbers above.

7 Jellyfish, "2025 State of Engineering Management Report", Jul 2025.

About Encore

Encore is a development platform for the AI era, built for a world where AI multiplies software delivery 10-100x rather than making each developer slightly faster.

In a traditional Terraform-based workflow, much of the infrastructure code cannot be meaningfully validated until it is applied to a real cloud environment, so mistakes surface late, in staging or production, where they are slower and more expensive to fix. AI does not solve this by producing Terraform at roughly human quality: if the error rate stays the same while change volume grows 10-100x, the issues reaching the deployment loop grow 10-100x with it. At that scale, generated infrastructure would need to be effectively perfect rather than merely human-level.

Encore changes the model instead. Developers and AI agents declare infrastructure requirements, such as databases, queues, and scheduled jobs, directly in ordinary TypeScript or Go, and Encore derives the configuration, permissions, environments, and deployment setup from those declarations. The same application model runs locally, in isolated preview environments, and in production, so changes are validated with real infrastructure before they reach the production loop. The framework is open source, and the cloud resources are standard services in your own AWS or GCP account.

WHAT IT REMOVES	WHAT IT ACCELERATES	WHAT IT PROTECTS
Hand-written IaC and its maintenance	Idea to production for new services	Credentials and infra state outside the repo
Environment setup and drift	Preview environments per pull request	Least-privilege defaults on every resource
Per-service CI/CD and observability wiring	AI-assisted development inside guardrails	Full audit trail per resource and deploy

How to run the pilot

- → Pick one upcoming service. Build it on Encore alongside your existing stack. No migration required.
- → Measure the DORA four plus review-queue time against your baseline for one quarter.
- → Bring the deltas to the chapter 7 memo, and expand only if the numbers argue for it. In our experience they do.

> "Encore is our foundation for all new development. Since adopting it, we've seen a 2–3x increase in development speed and 90% shorter project lead times. What used to take days or weeks of back-and-forth between developers and infra teams is now automated and completed in minutes."
>
> **Josef Sima, Engineering Director**
>
> GROUPON

26% → 57%

LIKELIHOOD OF REALIZING AI VELOCITY GAINS, LOW → MODERATE CD AUTOMATION (HARNESS, 2025)

$200k+

ANNUAL SAVINGS ON HEADCOUNT COSTS REPORTED IN CARLA'S CASE STUDY (10X ROI)

90%

SHORTER PROJECT LEAD TIMES REPORTED BY GROUPON AFTER ADOPTING ENCORE

Run the pilot

Pilot one service for one quarter, measured against your own baseline.

Get started free

Book a demo