encore

The AI-Ready Infrastructure Playbook

Code moves at AI speed. Infrastructure doesn't. A guide for engineering leaders.

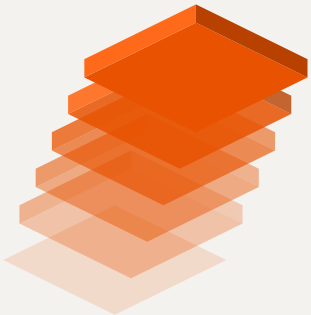


Table of Contents

Executive Summary

AI coding tools have made developers measurably faster at producing code, but software is not reaching production faster. Research across 2024 to 2026 shows the same pattern from every angle: individual output is up sharply, while delivery speed has improved only marginally and delivery stability has declined.[1]

Code generation was one stage of the software lifecycle, and AI compressed it, while the stages around it, review, environments, provisioning, deployment, and operations, still run at their old speed. 67% of organizations now report that development is moving ahead of infrastructure in AI adoption, and 93% have experienced at least one AI-caused infrastructure incident.[2]

This playbook examines where the gap comes from, what current research says AI can and cannot do in infrastructure, and what infrastructure that keeps pace with AI-speed development looks like. It closes with a readiness assessment you can run against your own organization.

[1] DORA, "Accelerate State of DevOps Report", 2024; DORA, "State of AI-assisted Software Development", 2025. dora.dev

[2] Spacelift, "The Infrastructure Automation Report 2026: The AI Readiness Gap", n=406, fielded by Panterra Group. spacelift.io/infrastructure-automation-survey-2026

1 Why Faster Coding Hasn't Meant Faster Delivery

Two out of three software firms have rolled out generative AI tools. The typical result is a 10 to 15% productivity boost, and Bain's assessment is blunt: the time saved often isn't redirected toward higher-value work. Companies that pair AI with end-to-end process transformation report 25 to 30%.[3]

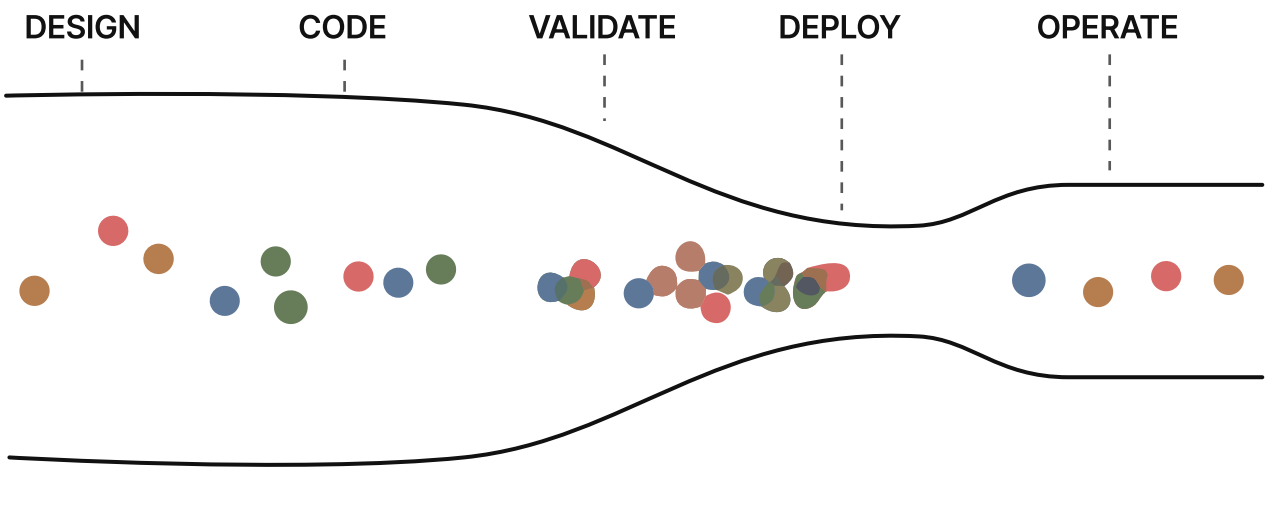
The gap between those two numbers comes from everything around the tooling. Writing and testing code accounts for only 25 to 35% of the time from idea to launch,[3] so compressing that one stage does not speed up the pipeline; it moves the queue downstream.

Where the queue forms

Telemetry from 8.1 million pull requests across 4,800 organizations shows what happens after the code is written: AI-generated pull requests wait 4.6x longer for their first review than human-written ones, and their acceptance rate is 32.7%, versus 84.4% for manual PRs.[4] The time saved in writing the code is absorbed by the stages that follow it.

FIGURE 01 — WHERE WORK ACCUMULATES

DESIGN → OPERATE



The stages after code generation still run at their previous speed, so work accumulates before deployment. Illustrative, not to scale.

Developers feel both sides of this at once. In Atlassian's 2025 survey, 68% of developers report saving 10 or more hours per week with AI. Half of them lose 10 or more hours per week to organizational friction: waiting on environments, hunting for information, and context switching.[5] The same developers who gain time from AI lose it again to the delivery system around them.

WHAT THE INCIDENT DATA SHOWS

72% of organizations have had at least one production incident caused by AI-generated code, and 45% of deployments involving AI-generated code lead to problems. Only 6% of teams describe their delivery process as fully automated.[6]

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

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

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

6 Harness, "The State of AI in Software Engineering 2025", Coleman Parkes survey of 900 engineers and leaders (US/UK/FR/DE), fielded Aug 2025. harness.io

2 Infrastructure Is Falling Behind

After the review queue, the next place AI-generated code runs into trouble is infrastructure. In a 2026 survey of 406 IT and platform engineering leaders, 67% report that development is moving ahead of infrastructure in AI adoption, and the consequences are already measurable: 93% of organizations have experienced at least one AI-caused infrastructure incident.[2]

FIGURE 02 — THE INFRASTRUCTURE GAP

2026 SURVEY, N=406

93%	67%	33%
HAD AN AI-CAUSED INFRA INCIDENT	DEV IS AHEAD OF INFRASTRUCTURE	WOULD SHIP AI HCL UNREVIEWED

Survey of 406 IT and platform engineering leaders at organizations of 250+ employees, fielded April 2026.[2]

The same survey shows how the gap is being papered over: 78% of teams use AI to generate infrastructure code without thorough review, and only 19% have governance foundations for AI in place. Teams are pushing AI-speed changes through infrastructure built for human-speed change control.[2]

The layer underneath was already strained

Infrastructure teams were behind before AI arrived. 35% of teams tie configuration drift to multiple costly production incidents in the past year. 90% agree their infrastructure orchestration falls short of what they need, and only 8% report managing infrastructure as code at scale without notable issues.[7] AI did not create this gap, but it has made the gap expensive.

DORA's research puts the two-year arc plainly. In 2024, every 25% increase in AI adoption was accompanied by an estimated 1.5% decrease in delivery throughput and a 7.2% decrease in delivery stability. By 2025, throughput had recovered, but the instability finding persisted, and the report's central conclusion changed shape: AI acts as an amplifier of the system around it. Organizations with strong platforms convert AI speed into delivery performance. Organizations without them convert it into incidents.[1]

WHY THIS LANDS ON THE CTO'S DESK

Unplanned downtime now costs Global 2000 companies an estimated $600B per year, roughly $15,000 per minute, which puts the infrastructure gap squarely on the P&L.[8]

7 Firefly, "State of IaC", 2025 and 2026 editions, and "The Bad IaC Tax", 2026. firefly.ai

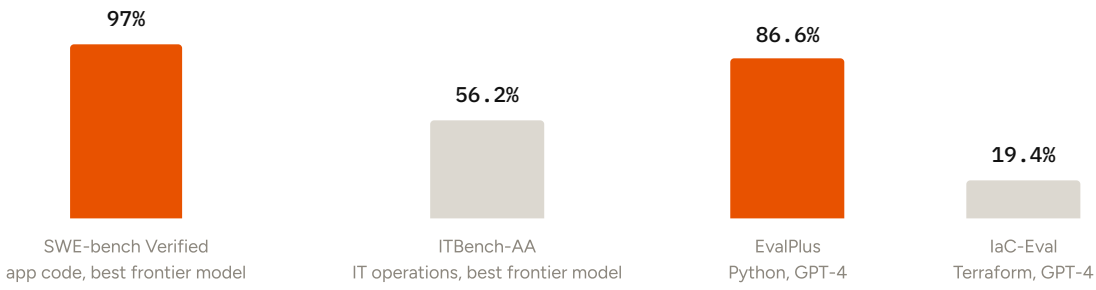
8 Splunk (Cisco) with Oxford Economics, "The Hidden Costs of Downtime 2026: A $600 Billion Wake-Up Call", May 2026. splunk.com

3 What AI Can and Can't Do in Infrastructure

The obvious objection is that this gap is temporary: models keep improving, so today's coding agents will operate infrastructure tomorrow. Scores are rising, but they trail application code by a wide margin, and the bar differs in kind: a coding agent's mistakes are caught by tests and review before they run, while an infrastructure agent's mistakes run in production, immediately.

On SWE-bench Verified, the standard benchmark for resolving real application-code issues, the best frontier models now score between roughly 93% and 97%.[9] On ITBench-AA, the 2026 leaderboard for agentic IT operations built on IBM Research's peer-reviewed ITBench, the best frontier model reaches 56.2% and most remain below 50%; IBM's original study measured 13.8% on SRE diagnosis.[10]

FIGURE 03 — SAME MODELS, APPLICATION CODE VS. INFRASTRUCTURE



Application code (orange) vs. infrastructure (gray). Sources: SWE-bench Verified leaderboards and Artificial Analysis ITBench-AA, Jul 2026; IaC-Eval, NeurIPS 2024.

The gap holds within a single model: GPT-4 passes 86.6% of tasks on EvalPlus, the standard Python benchmark, and 19.4% on IaC-Eval, its Terraform equivalent.[11] Nor is the gap a quirk of HCL syntax: on SWE-InfraBench, Amazon's benchmark of realistic changes to AWS CDK projects, where infrastructure is expressed in TypeScript and Python, the best model succeeds on 34% of single attempts.[12] A 2025 study of LLM-generated infrastructure code names the underlying pattern the Correctness-Congruence Gap: models have become proficient coders but remain limited architects. They produce syntactically valid resources that don't match the intent of the system around them.[13]

What AI handles well, and what it doesn't

WHAT AI DOES WELL TODAY	WHERE AI MEASURABLY FAILS
Generating and refactoring application code	Provisioning correct infrastructure for the code it wrote
Scaffolding new services and endpoints	Diagnosing and resolving live operational incidents
Writing tests and documentation	Keeping environments, networking, and IAM consistent
Translating requirements into business logic	Understanding system-level intent across resources

9 SWE-bench Verified leaderboards (vals.ai, llm-stats.com), July 2026. Top reported score 97.0% (Jul 22, 2026).

10 IBM Research, "ITBench: Evaluating AI Agents across Diverse Real-World IT Automation Tasks", ICML 2025 (arXiv:2502.05352); Artificial Analysis, ITBench-AA leaderboard, 2026.

11 Kon et al., "IaC-Eval: A Code Generation Benchmark for Cloud Infrastructure-as-Code Programs", NeurIPS 2024. 458 scenarios.

12 Amazon Web Services, "SWE-InfraBench: Evaluating Language Models on Cloud Infrastructure Code", NeurIPS 2025 workshop.

13 Nekrasov et al., "IaC Generation with LLMs: An Error Taxonomy and A Study on Configuration Knowledge Injection", arXiv:2512.14792, 2025.

Two ways to respond

These results leave engineering organizations with two options. The first is to keep humans in the loop for every infrastructure change and accept that infrastructure remains the rate limiter on everything AI produces upstream. That is the current default, and the review-queue data in chapter 1 shows what it costs. It also degrades as AI adoption succeeds: human-written infrastructure fails at some rate too, and the production loop absorbs those failures today only because human throughput caps their volume. Multiply change throughput by ten and the same per-change quality delivers ten times the failures to production, so matching human quality is the wrong bar once volume is the goal.

The second is to change what AI has to touch. If infrastructure requirements are expressed inside the application code, as typed, declarative statements of what the service needs, then AI tools operate entirely in the domain where they measurably perform: application code. The platform derives provisioning, environments, and configuration from those declarations, so there is no Terraform layer left for anyone, human or agent, to write.

THE DESIGN PRINCIPLE

Rather than sending agents into the layer where the best frontier model scores 56%, restructure the layer so the work happens where the same models score up to 97%. Infrastructure becomes an output of the code rather than a second codebase to maintain.

A growing set of platforms, Encore among them, already operate this way: the application declares databases, queues, caches, and APIs in ordinary TypeScript or Go, and provisioning, deployment, and observability are derived from static analysis of the code. Chapter 4 examines why the current standard, infrastructure as code, is under strain from more directions than AI alone, and chapter 5 lays out the capabilities that define the derived model.

> "A high-quality platform amplifies the effects of AI adoption on organizational performance. The positive impact of AI on organizational performance is strong when platform quality is high."
>
> DORA Research
>
> 2025 STATE OF AI-ASSISTED SOFTWARE DEVELOPMENT

4 The State of Infrastructure as Code

Infrastructure as code was the last generation's answer to infrastructure at scale, and it earned its place: versioned, reviewable, repeatable. Two things have changed since: the operational load has outgrown the model, and the ecosystem around its dominant tool has destabilized.

What IaC costs to maintain

Adoption is nearly universal, at 89% of organizations, but only 6% report complete IaC coverage of their cloud. Industry estimates put up to 40% of cloud resources outside IaC management entirely, with roughly a third of codified resources drifted from their definitions. 48% of teams make out-of-band production changes multiple times per week, which means the code and the cloud disagree by design.[7] The talent to fix this is scarce: 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.[14]

What happened to Terraform's ecosystem

FIGURE 04 — THE IAC ECOSYSTEM, 2023 TO 2026 PUBLIC RECORD

AUG 2023	HashiCorp moves Terraform from open source to the Business Source License. The community forks it as OpenTofu under the Linux Foundation.
2024	IBM announces its acquisition of HashiCorp. Regulatory reviews in the US and UK delay the close.
FEB 2025	IBM completes the acquisition at a $6.4B enterprise value. Terraform's roadmap now runs through IBM.[15]
2025	OpenTofu joins the CNCF sandbox and passes 10 million downloads. GitLab 18.0 removes its Terraform CI/CD templates, citing the license. Fidelity Investments details migrating 50,000+ Terraform state files to OpenTofu.[16]
2026	78% of teams use AI to generate infrastructure code, and one third of infrastructure teams would apply AI-generated HCL to production without review.[2]

License change, acquisition, fork adoption, and the current state of AI-generated infrastructure code.

What came after "infrastructure as code"

The step beyond IaC is infrastructure derived from the application code itself. The earliest tools in this category leaned on opaque inference, where an SDK guessed what to provision and operators lost visibility into what existed and why. The designs that have taken hold since pair inference with explicitness: infrastructure requirements are declared in the application code as typed resources, the platform provisions them, and operators keep full visibility into the result.[17] In the IaC model, someone, human or agent, writes and maintains a second codebase that the benchmarks in chapter 3 show AI handles poorly. The declaration model removes that codebase entirely, which is what lets this layer keep pace with AI-generated change.

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

15 IBM Newsroom, "IBM Completes Acquisition of HashiCorp", Feb 27, 2025.

16 CNCF, Apr 2025; OpenTofu blog, "OpenTofu 1.10.0", Jun 2025, and "Fidelity Investments Shares Its Migration Story", Oct 2025; GitLab, "A guide to the breaking changes in GitLab 18.0", 2025.

17 Joab Jackson, "Infrastructure From Code: What Went Wrong", The New Stack, Jun 2025.

5 What Defines AI-Ready Infrastructure

DORA's 2025 research identifies internal platform quality as the factor that determines whether AI adoption translates into organizational performance.[1] The question for engineering leaders is what, concretely, such a platform must do when a large share of code is machine-generated. Four capabilities determine whether infrastructure keeps pace with that volume.

FIGURE 05 — WHAT THE PLATFORM ABSORBS

ENCORE

Infrastructure derived from application code PLATFORM

Provisioning	Environments	Guardrails	Observability
New resources ship with the code that needs them, with no separate definition to keep in sync.	Local, preview and production derived from one source, identical by construction.	Typed resources and least-privilege defaults. Unsafe configuration fails at compile time.	Tracing, metrics and logs generated automatically, for humans and agents alike.

Each capability removes a category of work rather than accelerating it.

Review queues shrink when the platform, not the reviewer, guarantees that a change cannot reach production with an unprovisioned dependency or an untraced endpoint. That is what converts AI-generated volume from a liability back into throughput.

What this means in practice

FOR DEVELOPERS AND AGENTS

- Work stays in application code, the domain where AI performs
- Declaring a database or queue is a line of typed code
- Instant local environments that match production
- No context switch into Terraform, YAML, or consoles

FOR PLATFORM TEAMS

- One source of truth, no drift between code and cloud
- Guardrails defined once, enforced on every change
- Full visibility into what exists and why
- Support burden shifts from tickets to policy

FOR THE ORGANIZATION

- AI throughput converts to delivery, not queue depth
- Delivery stability holds as volume grows
- Lower exposure to IaC talent scarcity
- Audit trail on every provisioned resource

> "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

PLATFORM AUTOMATION DOUBLES THE ODDS

Harness's 2025 research found that organizations moving from low to moderate continuous-delivery automation more than double their likelihood of realizing velocity gains from AI, from 26% to 57%.[6] The return on the AI investment scales with the platform underneath it.

6 Security and Governance in the Agent Era

AI tools read the repository. In a conventional backend setup, the repository contains .env files, Terraform state with cloud account details, and connection strings for production databases. Measured telemetry shows what follows: commits assisted by a leading AI coding agent showed a 3.2% secret-leak rate, more than double the 1.5% baseline across all public GitHub commits, and 2025 saw the largest single-year jump in exposed secrets on record.[18]

Governance has not kept up with usage. 34% of organizations report that more than 60% of their code is now AI-generated, while only 18% have policies governing the use of AI coding assistants.[19] ISACA puts formal, comprehensive AI policy coverage at 28% of organizations worldwide; the Cloud Security Alliance puts comprehensive AI security governance at 26%.[20] Adoption is running roughly three times ahead of policy.

WHEN IT GOES WRONG AT SCALE

In March 2026, Amazon's retail site suffered a series of outages in one week. By Amazon's own account, one incident involved an engineer following inaccurate advice an AI tool inferred from an outdated internal wiki. Internal documents reported by the Financial Times and CNBC described a "trend of incidents" tied to "Gen-AI assisted changes"; Amazon disputes parts of that reporting and says it has updated internal guidance.[21]

Shrink the attack surface

The structural alternative is to reduce what AI tools can see. When credentials, infrastructure state, and access policies live in the platform rather than the repository, AI tools see business logic and typed resource declarations, nothing else. The EU AI Act becomes fully applicable in August 2026, and auditors have issued no guidance on AI-generated code under SOC 2 or ISO 27001, so "the AI never had access to production credentials" is the strongest control argument available.[22]

FIGURE 06 — WHAT AI TOOLS CAN REACH — REPOSITORY VS PLATFORM

WHAT AI CAN ACCESS	TRADITIONAL SETUP	PLATFORM-MANAGED SETUP
Secrets	.env files, API keys, connection strings in the repo	Managed by the platform. Not in the repo
Infrastructure config	Terraform and state files with cloud account details	Derived from application code automatically
Deployment config	Dockerfiles, CI/CD pipelines, deploy scripts	Handled by the platform
Application code	Business logic plus everything above	Business logic only

What an AI coding tool can read in each setup.

18 GitGuardian, "The State of Secrets Sprawl 2026" (Claude Code-assisted commits: 3.2% vs 1.5% baseline across all public GitHub commits). gitguardian.com

19 Checkmarx, "Future of Application Security in the Era of AI", Aug 2025, 1,500+ CISOs, AppSec managers and developers.

20 ISACA, 2025 AI Pulse Poll, 3,029 digital trust professionals worldwide; Cloud Security Alliance with Google Cloud, "The State of AI Security and Governance", Dec 2025, n=300.

21 Amazon, aboutamazon.com, Mar 2026; Fortune, Mar 12, 2026; Financial Times and CNBC reporting; Wharton Accountable AI Lab, Apr 2026.

22 Regulation (EU) 2024/1689, Art. 113; European Commission AI Act implementation timeline.

7 Assessing Your Infrastructure Readiness

Score each area out of four, one point per capability you have in place. Few organizations score full marks, and where the low scores cluster matters more than the total.

FIGURE 07 — READINESS SCORECARD

SCORE EACH AREA OUT OF FOUR

Provisioning ticketless resources, every one traceable to code	____ / 4
Environments local, preview and production identical; preview per PR	____ / 4
Deployment no manual steps; failure rate holding as volume grows	____ / 4
Observability traced by default; debuggable without asking anyone	____ / 4
AI governance written policy; credentials out of reach by construction	____ / 4

One point per capability in place, twenty available.

0–7	8–13	14–20
Exposed	**Partial**	**AI-ready**
AI is ahead of the controls	Gains real, leaking through manual stages	The platform converts velocity into delivery

HOW TO READ YOUR ANSWERS

Low scores in provisioning and environments point at the infrastructure gap this paper describes, and they are the ones that compound as AI-generated volume grows. A low governance score is exposure that grows with every percentage point of AI-written code. Anything low in deployment or observability usually means the queue from chapter 1 is forming in your pipeline too.

8 Getting Started

Closing the infrastructure gap does not require a migration program. The pattern that works starts where the risk is lowest and the contrast is most visible.

FIGURE 08 — A QUARTER, NOT A PROGRAMME

ONE SERVICE, MEASURED

STEP 01

Score

Score the five areas in figure 07 with your platform and engineering leads, and agree which gaps cost most.

STEP 02

Pilot

One upcoming service on derived infrastructure, alongside the existing stack. No migration.

STEP 03

Measure

Lead time, deploy frequency, change failure rate and review wait, against today's baseline.

STEP 04

Expand

Move services as the data justifies it. New work takes the new model by default first.

WEEK 0 — CURRENT STATE

WEEK 12 — DECIDE ON EVIDENCE

A single service carries the pilot. Metrics come from your own baseline rather than a vendor benchmark.

FIGURE 09 — WHAT TEAMS REPORT

CUSTOMER-REPORTED

26% → 57%

AI GAINS REALISED, LOW TO MODERATE CD AUTOMATION[6]

90%

SHORTER LEAD TIMES, REPORTED BY GROUPON

2–3x

FASTER DEVELOPMENT, REPORTED BY GROUPON

Outcomes reported by Encore customers in published case studies, alongside the Harness figure for delivery automation.

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, application code and infrastructure are developed separately. Developers or AI tools write Terraform, but much of it cannot be meaningfully validated until it is applied to a real cloud environment. The result is a production-validation loop: infrastructure mistakes surface late, after deployment to staging or production, where they are slower, riskier, and more expensive to fix. This is already a problem with human-written Terraform, and AI does not solve it by producing Terraform at roughly human quality. If the error rate stays the same while the number of changes grows 10-100x, the number of infrastructure issues reaching the deployment loop grows 10-100x with it, which is operationally unsustainable. At that scale, generated infrastructure would need to be effectively perfect rather than merely as good as a human's.

DECLARE IN CODE

- Type-safe APIs, databases, queues, caches, and cron jobs
- Standard language patterns, no DSLs
- Infrastructure semantics checked at compile time

DEPLOY ANYWHERE

- AWS or GCP, in your own cloud account
- Preview environments per pull request
- Production deployments automated end to end

OBSERVE EVERYTHING

- Distributed tracing with zero configuration
- Metrics, logging, and service catalog built in
- Architecture diagrams generated from the code

How Encore changes the model

Encore changes the model instead of generating Terraform faster. Developers and AI agents declare infrastructure requirements, such as databases, queues, and scheduled jobs, directly in ordinary TypeScript or Go, and Encore derives the infrastructure configuration, permissions, environments, and deployment setup from those declarations, provisioning the resources in your own AWS or GCP account.

Because the same application model runs locally, in isolated preview environments, and in production, changes are run and validated before they reach the production deployment loop. AI agents can test the real system, including its infrastructure dependencies, while they work, rather than generating configuration that is only proven when deployed.

The result is a delivery system designed for much higher throughput: AI focuses on application changes, and Encore makes those changes runnable and testable with real infrastructure before production.

> "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

Start building with Encore

See what your team ships when infrastructure moves at the same speed as the code.

Get started free

Book a demo