encore

Our code got faster. Our infrastructure did not.

Where the time goes now, why review does not catch it, and what to try next.

FROM THE AI-READY INFRASTRUCTURE PLAYBOOK

10 MIN

THE PROBLEM

Output is up, but nothing reaches customers faster.

Writing code was one stage of the work, and it is the one that sped up. Everything after it still runs at the old speed, and each request still needs a stack of infrastructure to run.

93%

of organisations have had at least one AI-caused infrastructure incident

45%

of deployments containing AI-generated code lead to problems

4.6x

longer wait for a first review on AI-authored pull requests

Infrastructure Automation Report 2026 (n=406) · State of AI in Software Engineering 2025 (n=900) · Software Engineering Benchmarks 2026 (8.1M PRs)

Who tells you that you got it wrong

APP CODE

```
10:02:04 run tests
10:02:06 2 failing
10:02:19 edit · run tests
10:02:21 1 failing
10:02:44 edit · run tests
10:02:46 passing
10:03:10 open pull request
```

Wrong twice, fixed twice, in 66 seconds.

INFRA CODE

```
10:02:04 terraform validate
10:02:07 valid
10:02:12 terraform plan
10:02:41 6 to add, 1 to change
10:04:00 open pull request
14:20:00 merged, applied
02:14:31 paging on-call
```

Every check passed. The pager went off 16 hours later.

You only find out in production, so an agent gets one attempt.

THE FIX

Give infrastructure the same loop, by deriving it from the code.

Encore is one platform built this way. Whoever makes the change can run it and see the result before production.

YOUR CODE & AGENTS

Typed declarations in TypeScript or Go

ENCORE

Provisions the infrastructure

YOUR CLOUD

Runs in your own AWS or GCP

INFRASTRUCTURE FROM APPLICATION CODE

What that looks like in practice

Resources are declared as typed objects in ordinary TypeScript or Go, and the platform derives provisioning, permissions, environments and tracing from them, with sensible defaults you can override in the dashboard.

```
const users = new SQLDatabase("users", { migrations: "./migrations" });
const signups = new Topic<SignupEvent>("signups", { deliveryGuarantee: "at-least-once" });
```

LOCAL

On your machine

A database and queue running locally, not mocked.

PREVIEW

One per pull request

Reviewers open the running change, not a plan.

PRODUCTION

Your own AWS or GCP

Same declarations, tracing already attached.

All three come from the same declarations, so a change runs and gets fixed before anyone reviews it.

WHERE WE STAND

How late do we find out today?

Four possible answers. Whichever one is true for us is the size of the problem, because the later we find out, the more AI-generated volume makes it hurt.

At build time	Rare. The loop is already closed.
In code review	A person reading intent. The expensive mistakes get through.
In staging, sometimes	Shared, slow to reset, and skipped when we are in a hurry.
In production	Gets worse the better AI adoption goes.

THE PROPOSAL

Build one new service this way, and compare the numbers

Pick a service we were going to build anyway and build it on a platform that derives infrastructure from application code, Encore for example. Everything we run today stays where it is, and nothing gets migrated. A sprint to build, then it runs on its own, and we decide on our own numbers.

A SPRINT	THEN	AFTER
Build the service	Leave it running in production	Compare

THE FOUR NUMBERS, TAKEN BEFORE WE START AND AGAIN ONCE IT HAS RUN

Lead time	Review wait	Change failure rate	Mistakes caught late
Commit to running in production	Opened to first review	Deploys needing a fix	How many reached staging or production

Sources, benchmarks and the full readiness diagnostic are in the report.

Read the full report