Build a better Loop: A guide to platform engineering (in the age of AI)

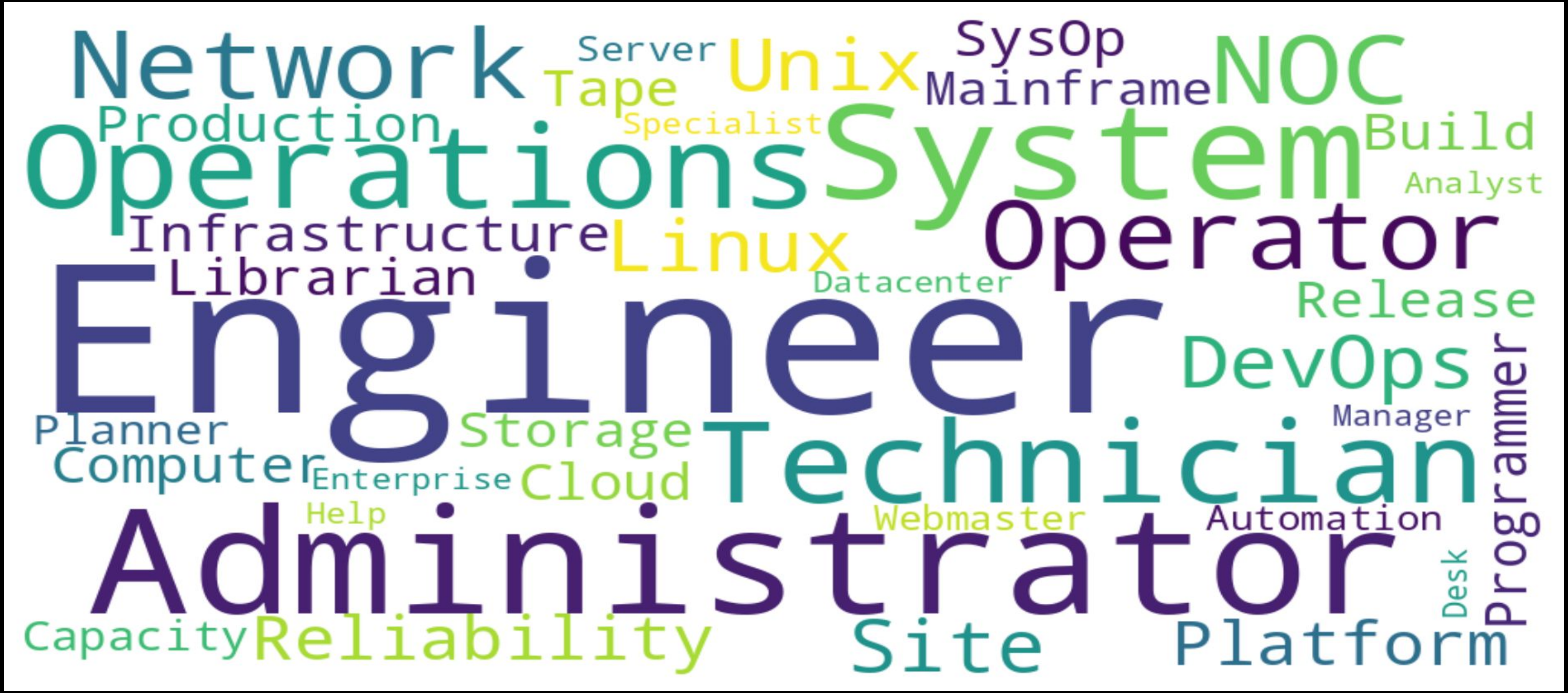
Arthur Freyman

Director of Engineering, ZipRecruiter

What We'll Cover

- The pattern that keeps repeating
- What platform engineering is today — and the product discipline behind it
- Which of our assumptions AI just broke
- What the product becomes next

The Divide


Network
Server
Unix
SysOp
NOC
Tape
Mainframe
Production
Specialist
Build
Operations
System
Analyst
Operator
Infrastructure
Linux
Librarian
Datacenter
Engineer
Release
DevOps
Programmer
Manager
Planner
Storage
Technician
Computer
Enterprise
Cloud
Help
Webmaster
Automation
Administrator
Desk
Capacity
Reliability
Site
Platform

Frameworks Evolved Us

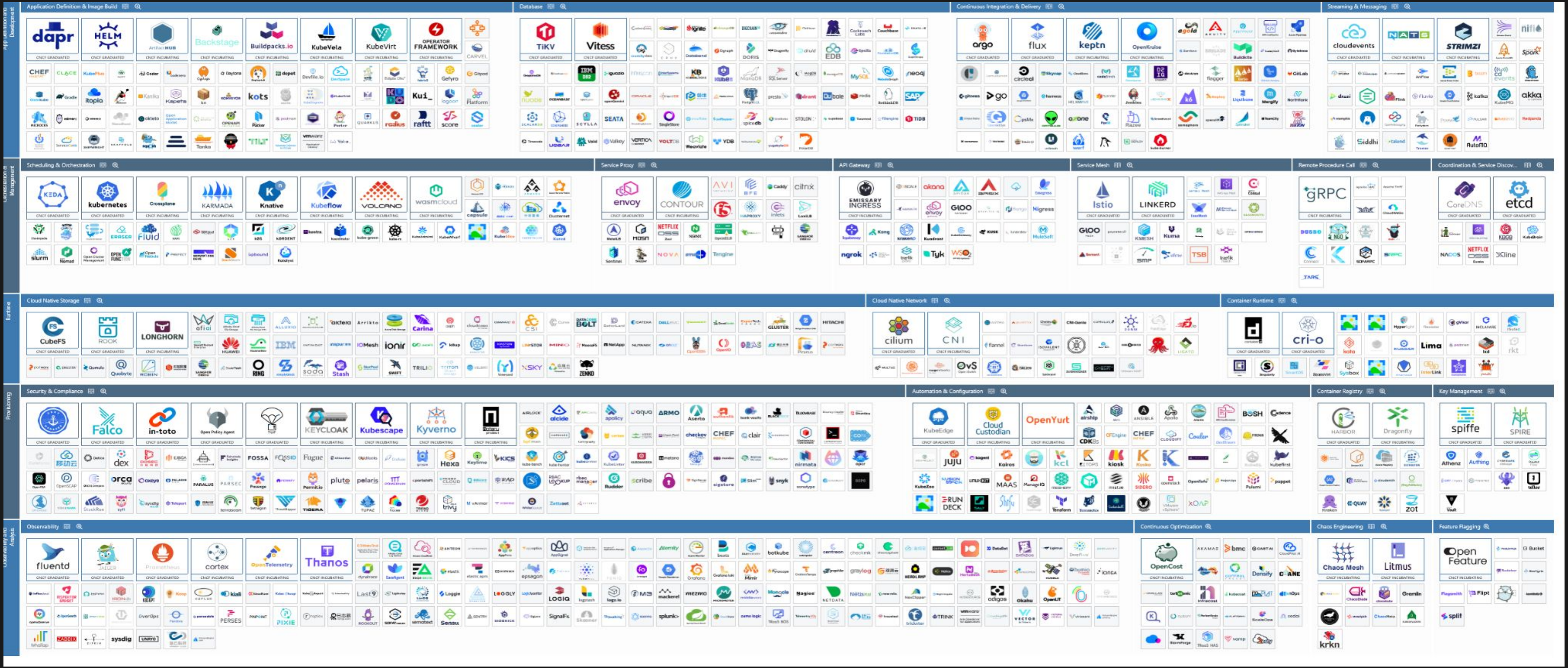
The job stayed the same. The frameworks got better.

DevOps → *"tighten the loop"* — gave us shared ownership

SRE → *"error budgets"* — gave us reliability as engineering

Platform Eng → *"product thinking"* — gave us the customer lens

Platform Engineering



What It's NOT

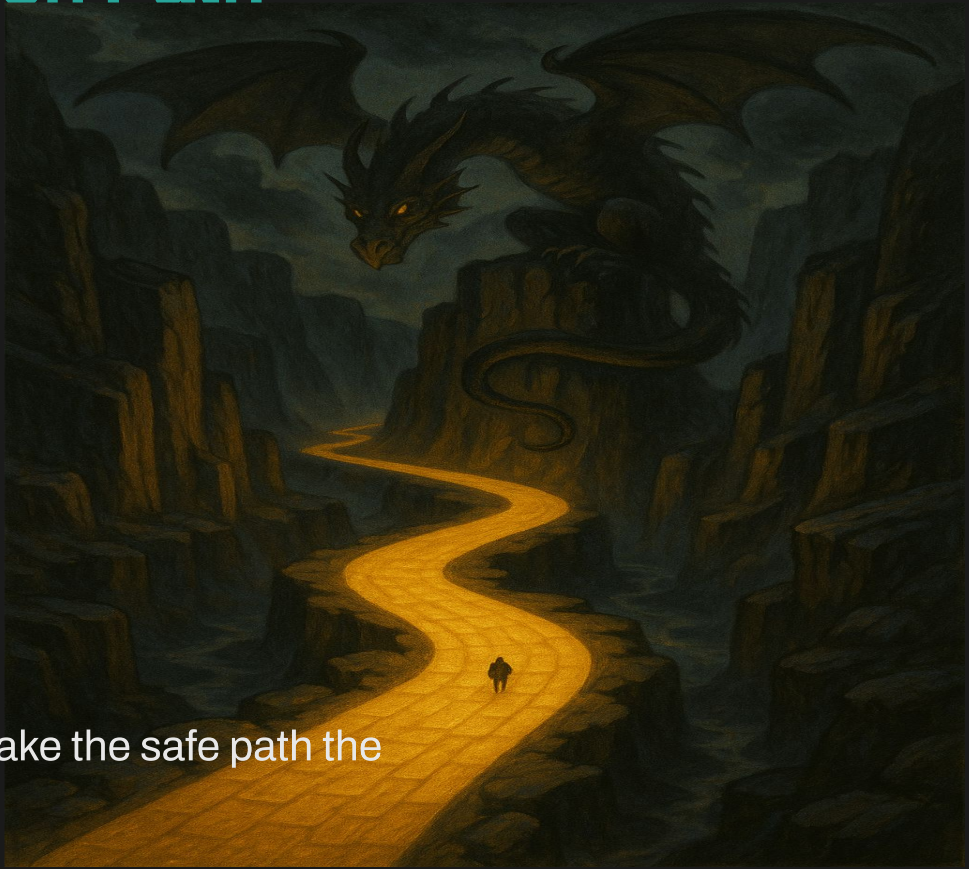
- It's NOT yaml / CRD engineering
- It's NOT Kubernetes
- It's NOT AWS
- Is it PaaS? Maybe.

It's Glue-as-a-Service (GaaS)

"A digital platform is a foundation of self-service APIs, tools, services, knowledge and support which are arranged as a compelling internal product. Autonomous delivery teams can make use of the platform to deliver product features at a higher pace, with reduced co-ordination."

— Evan Bottcher

The Golden Path



The platform's job: make the safe path the easy path.

What the Platform Actually Is

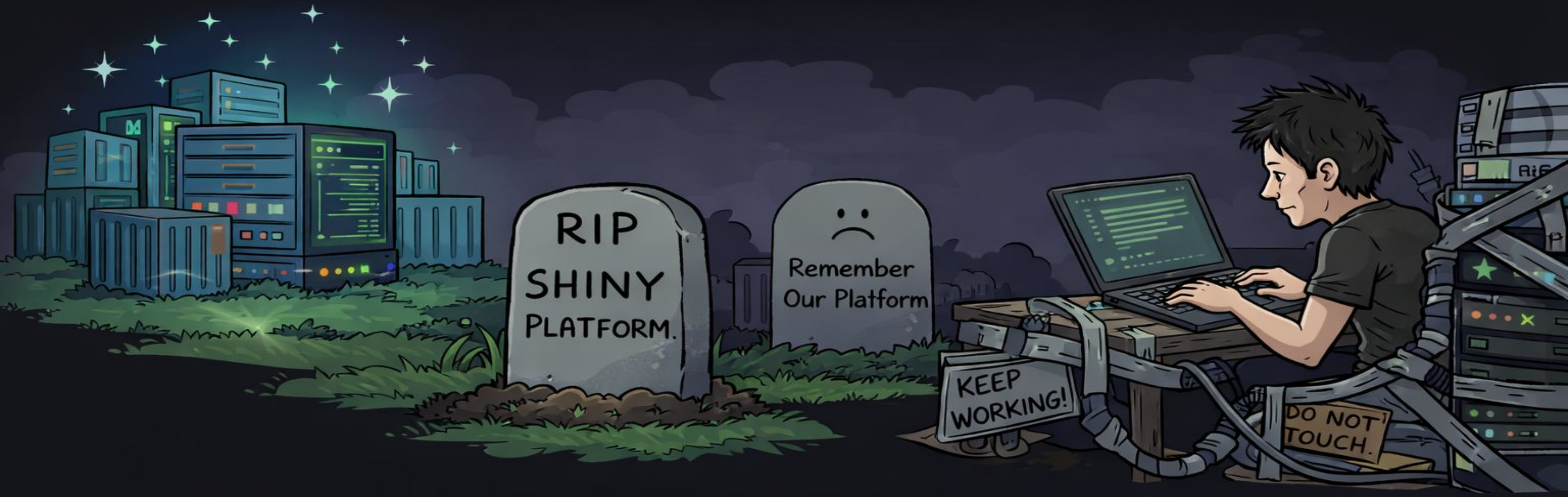
Golden path = the safe & fast path is the easy path

Platform = constraints + self-service + legibility

The interface is interchangeable: portal / CLI / chat / API

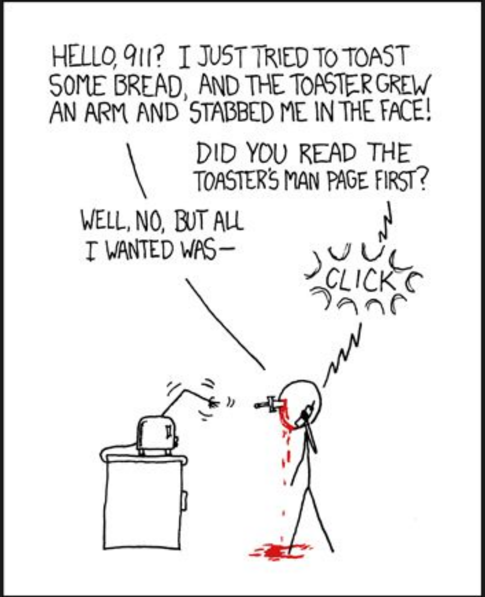
"In an agent world, the UI is incidental. The platform is the contract."

Nobody wanted to use it.



Understand Your Customer

- Measure: velocity + reliability as a control pair (DORA / SPACE / DevEx)
- Value stream mapping: do it with your customers, not to them
- "If your customers hate your product, that's not a training problem"


HELLO, 911? I JUST TRIED TO TOAST SOME BREAD, AND THE TOASTER GREW AN ARM AND STABBED ME IN THE FACE!
DID YOU READ THE TOASTER'S MAN PAGE FIRST?
WELL, NO, BUT ALL I WANTED WAS—
CLICK

Design for Adoption

- Cohesion: unified experience, consistency across surfaces
- Trade-offs: Freedom / Velocity / Maintainability — pick 2
- Offramps: the platform shouldn't be a prison
- Delightful to use — for humans

AI doesn't replace your platform team.

**It exposes that your platform team was never doing
what it should have been doing.**

"When cloud came along, the right move wasn't to perfect your VMware infrastructure."

Build the next generation, not the last one.

The Spec Gap

AI builds exactly what you specify.
Including your mistakes.

Your Customer Is Changing

Today: Developers using AI as a tool (Copilot, Claude, Cursor)

Emerging: AI agents as direct consumers of your platform

Reality: Agents are great customers AND terrible customers

"AI didn't change what we ship. It changed what's expensive."

- Implementation got cheap → decisions and coordination expensive
- Code got cheap → verification expensive
- Observability got cheap → joinable telemetry expensive

“Any improvements made anywhere besides the bottleneck are an illusion”

Eliyahu M. Goldratt

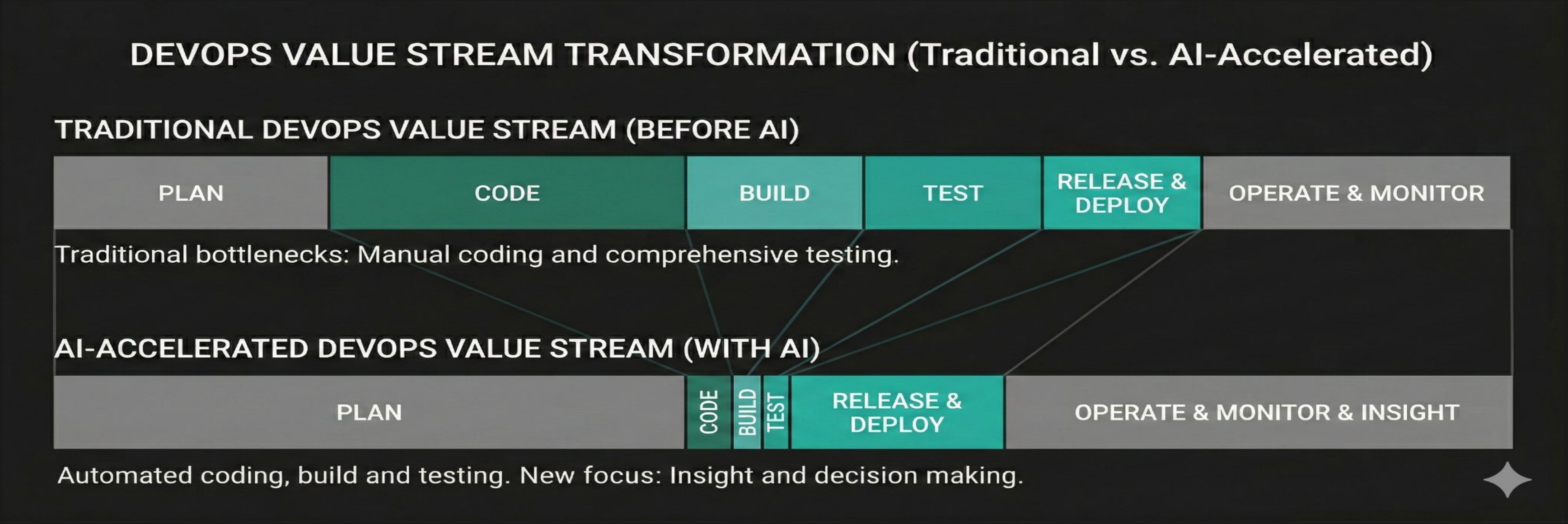
Prior 1: "Implementation Is the Bottleneck"

Was: "Implementation is the bottleneck"

Now: AI compresses implementation; lead time moves to decisions / dependencies / validation

Build for: The bottleneck moved. Are you still optimizing the old one?

Value Stream: Before & After AI


DEVOPS VALUE STREAM TRANSFORMATION (Traditional vs. AI-Accelerated)
TRADITIONAL DEVOPS VALUE STREAM (BEFORE AI)
PLAN
CODE
BUILD
TEST
RELEASE & DEPLOY
OPERATE & MONITOR
Traditional bottlenecks: Manual coding and comprehensive testing.
AI-ACCELERATED DEVOPS VALUE STREAM (WITH AI)
PLAN
CODE
BUILD
TEST
RELEASE & DEPLOY
OPERATE & MONITOR & INSIGHT
Automated coding, build and testing. New focus: Insight and decision making.

Same total lead time. The time just moved.

Prior 2: "Human Code Review Is Where Quality Happens"

Was: "Human code review is where quality happens"

Now: AI reviews most code; humans review critical changes, architecture, risk

Build for: Verification is the craft — tests + constraints + telemetry

The Verification Stack

Constraints / Policy
What must be true — guardrails as code

Tests / Contracts
Deterministic validation for non-deterministic generation

Telemetry / Legibility
Machine-readable state — the ceiling on automation

Prior 3: "Our Observability Is Good Enough"

Was: Good enough for humans who fill gaps with tribal knowledge

Now: AI/automation needs joinable state; garbage telemetry caps automation

Build for: Observability as machine-consumable control plane

Freedom / Velocity

Maintainability → the binding constraint

"The job used to be writing code."

"Now it's steering."

What We're Building Now

Surface	What It Means
Identity + ownership + metadata	Joinable reality — who owns what, how things connect
Constraints / policy	What must be true — discoverable, machine-readable, authoritative
Verification harnesses	Tests, SLO gates, contract checks
Legibility	Dependency graphs, cost attribution, blast radius
Workflow instrumentation	Agent funnel, intervention rate, stuck loops

*"Anyone who tells you they know what this looks like
in two years is selling something."*

"The divide survived mainframes, client-server, cloud, and DevOps."

"That divide has survived every technology shift so far."

Monday Actions

1. Audit your tests: can an agent verify correctness without a human?

2. Audit your legibility: starting from a service name, can an agent answer — who owns it, what depends on it, what SLOs govern it, what changed in the last 24 hours?

3. Map your value stream — map decisions, not code

The tools keep changing.
The job stays the same.