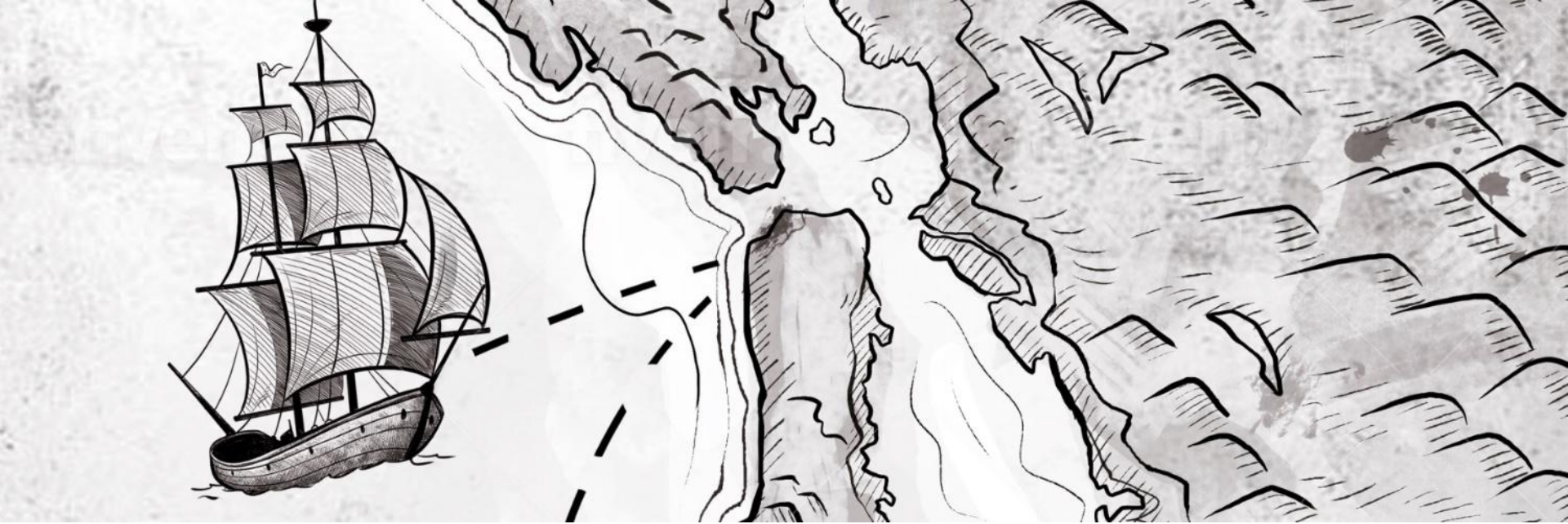
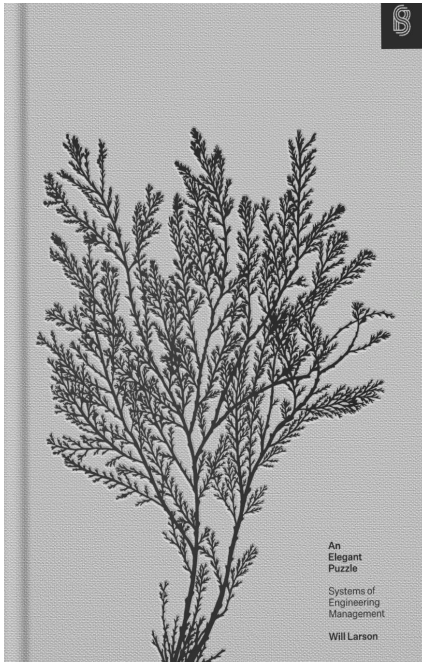


How we migrated from manual deploys to agent-driven development in six months

Will Larson

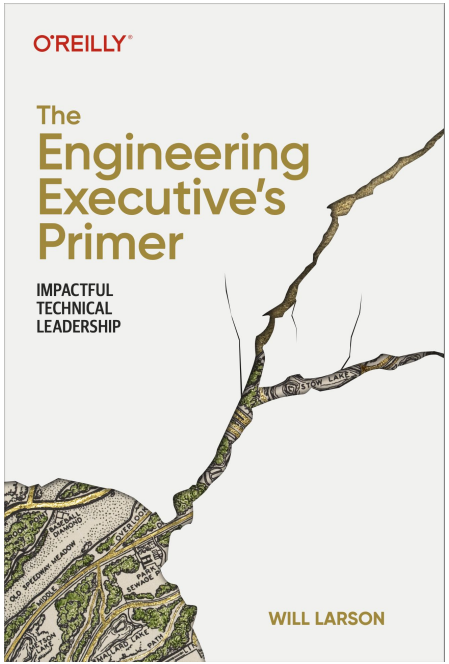
2026/9/15

The best way to pay: the modern, co-branded credit card platform powering programs for Shell, Fanatics, and many more.

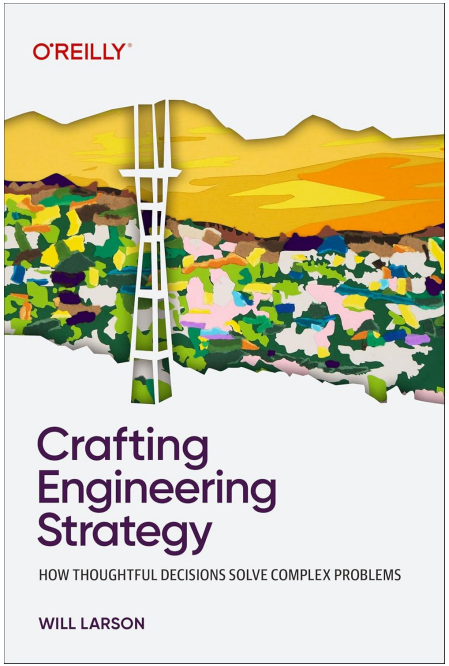


2019

2021



2024



2025

Conversation on AI adoption has anchored on:

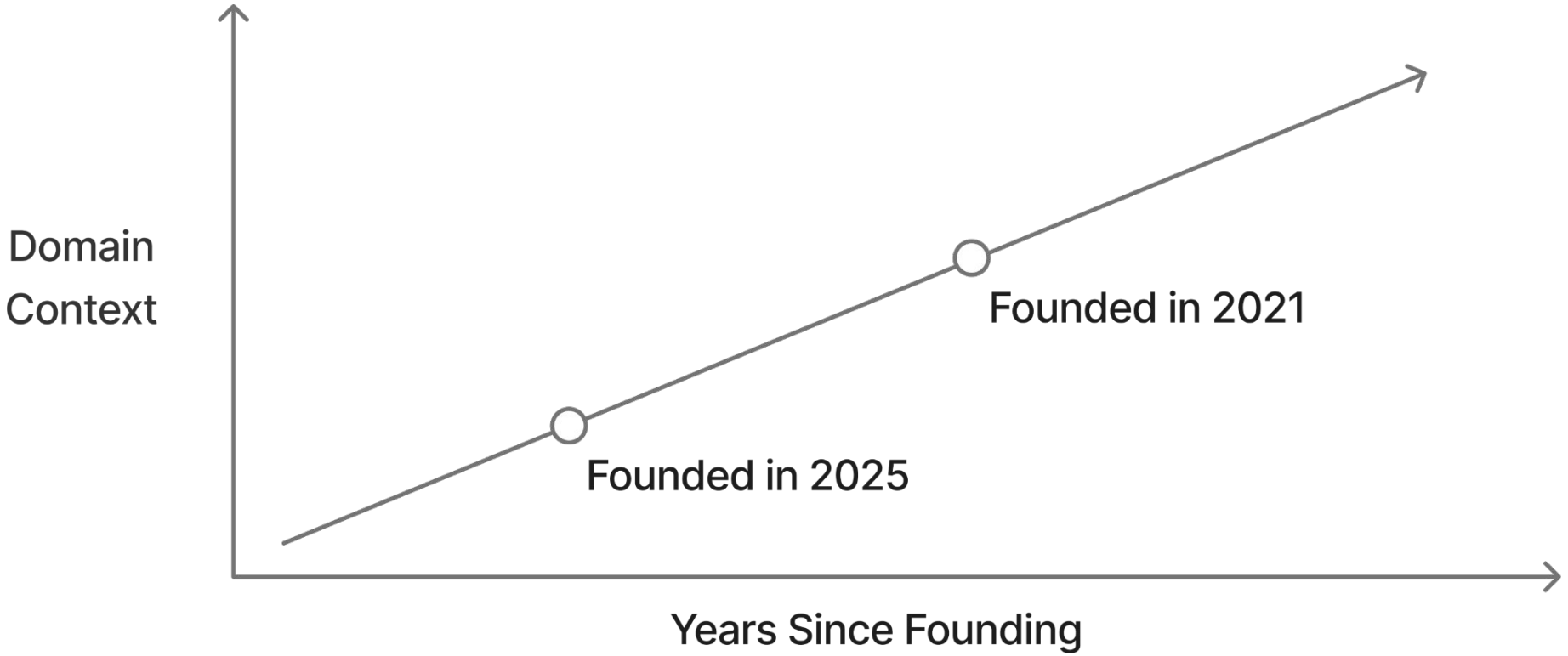
1. **Startups** founded in 2025+
2. **Established companies** with dedicated teams working on DX, Observability, etc

This talk is about **everyone else**.

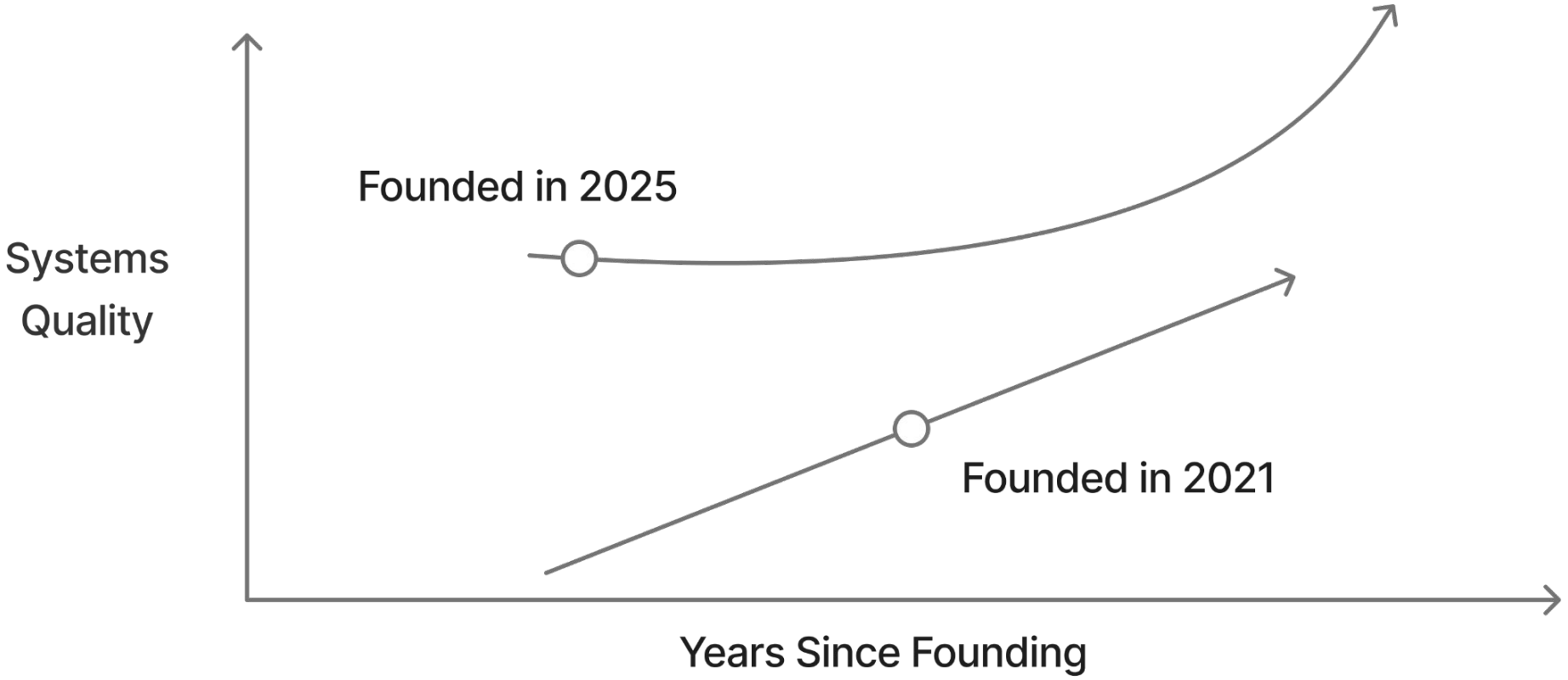
Thesis

Everyone else has 1-2 years to catch up before value of our domain context is outweighed by the limitations in our systems.

Domain context over time



Systems quality over time



Question

How do we (aka *everyone else*) catch up in those 1-2 years?

Answer

There’s no single answer, but I can talk about Imprint’s experience: migrate a lot.

Topics

1. How Imprint developed on Jan 1st
2. Ex 1: Migration to full CI/CD
3. Ex 2: Adoption of local harnesses
4. Ex 3: Adoption of remote harnesses
5. How we design migrations

(1) How Imprint Developed on Jan 1st

Our manual deployment steps:

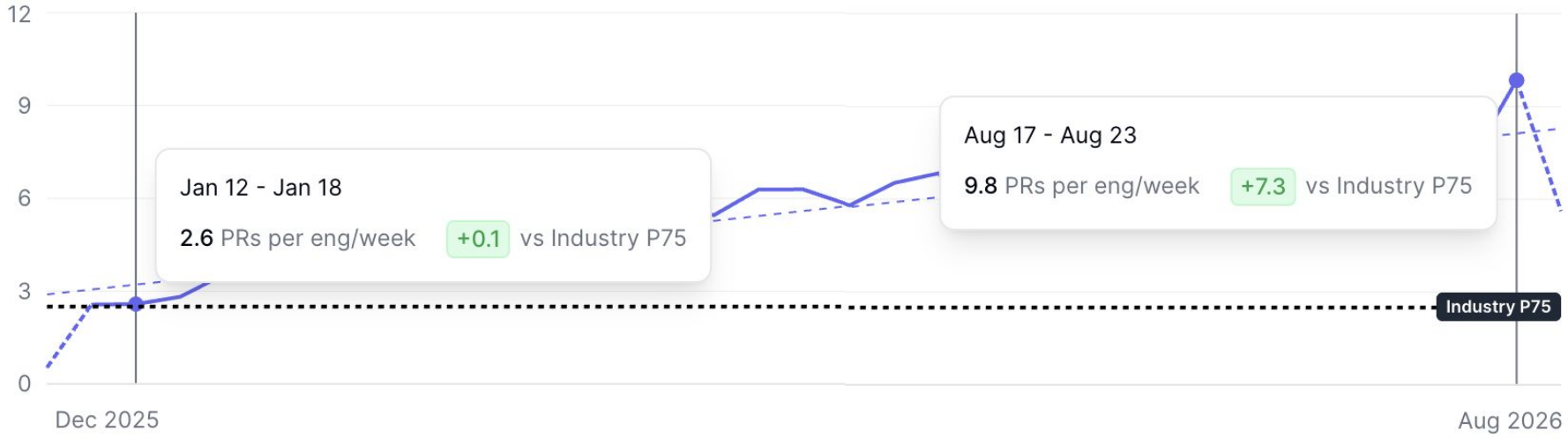
1. Pull request
2. Review
3. Merge
4. for env in envs:
    a. for service in services:
        i. Build service
        ii. Migrate service
5. Verify

Consequences of those deployment steps:

1. Required high attention to detail
2. Deploy’s risk was sum of prior errors
3. Making changes was scary
4. We spent a lot of time worrying
    a. (Also a lot of time avoiding)

(2) Ex 1: Migration to full CI/CD

PRs merged per engineer per week



Configuration-driven opt-in

```yaml
service:
  name: bridge

cicd_path:
  - stg
  - sbx
  - prd

cicd_env:
  synthetic_query: false
  mysql_auto_migrate: false
  regions:
    - us-east-1
```

Opt-in to `stg` only initially to validate for a new service

Reusable build

Opt-in database migration

```yaml
service:
  name: portal

cicd_path:
  - stg
  - ops-live

cicd_env:
  mysql_auto_migrate: true
  mysql_dbs:
    - db_name: portal
      migration_dir: portal/databases/migration/sql
  regions:
    - us-east-1
```

Opt-in canary-based deployment

```yaml
service:
  name: api
cicd_path:
  - stg
  - sbx
  - prd
cicd_env:
  mysql_auto_migrate: false
  argocd_timeout: 1200
  # Datadog monitor IDs for canary analysis. ALL must stay OK for 15 min.
  monitor_ids:
    - '267300537'   # [CANARY] Api Error Rate Elevated
    - '267300551'   # [CANARY] Api Canary Exists But Not Receiving Traffic (composite)
    - '267300554'   # [CANARY] Api Canary Panic Detected
    - '267294171'   # [CICD] ALB target 5xx error rate is above 5%
    - '267300272'   # [CICD] ALB 4xx volume is anomalously high
    - '271156648'   # [CANARY] Webhook Authentication Failures (403s)
  regions:
    - us-east-1
```

Our migration sequence:

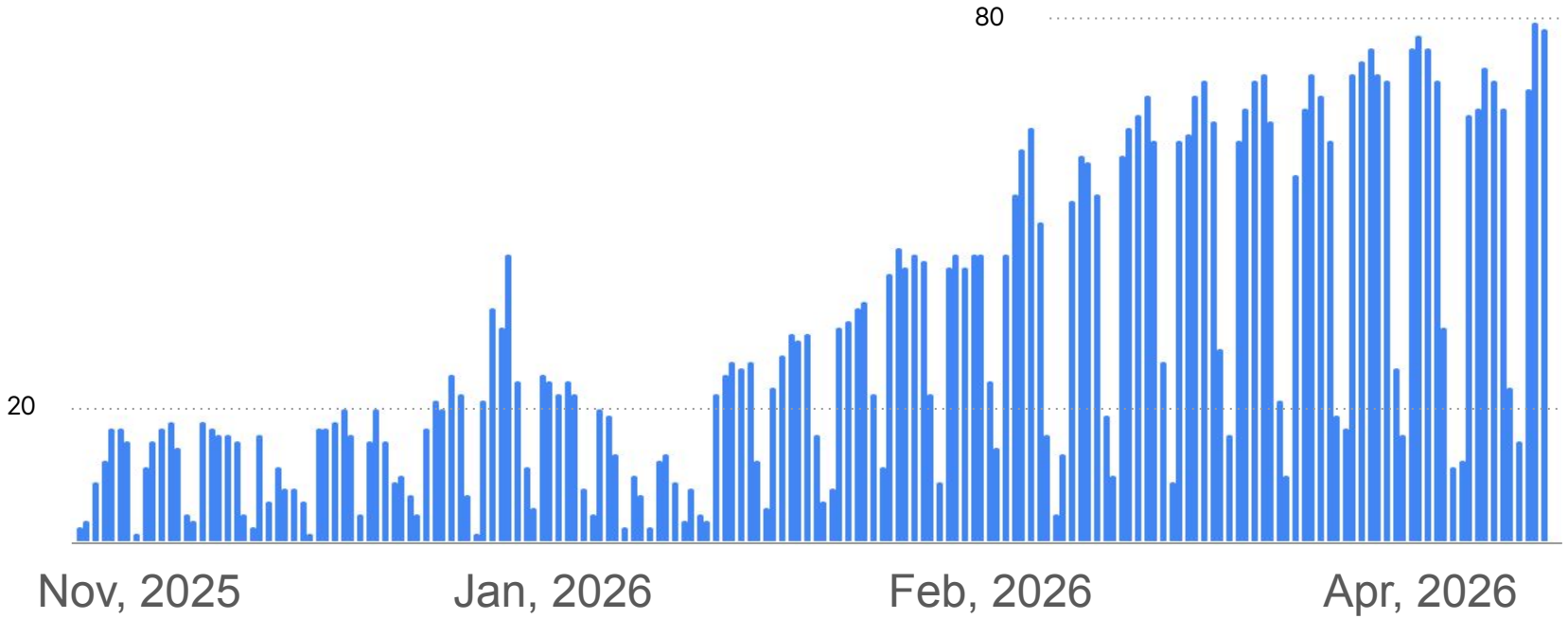
1. Opt-in from within monorepo
2. Migrated new, non-production service
3. 3 engineers moved everything in `stg` in <4 weeks
4. `prd` rollout was paced by per-service canary monitors (still in <4 weeks after `stg`)

Why this worked:

1. Zero new infrastructure
2. Go fast by derisk change (opt-in, canaries, …)
3. DevEx + Reliability
4. Working group ate a lot of pain (no distributing pain)
5. Risk of not moving was substantially higher

(3) Ex 2: Adoption of local harnesses

Daily Users of Local Harness (Claude, Codex or Cursor)



We mandated adoption and moved on.

~~We mandated adoption and moved on.~~

jk jk jk, no mandates

What we actually did was:

1. Usage dashboards
2. One golden path (Bedrock + Claude Code)
    a. Allowed others but no support
3. Explicit decision doc
4. #ai channel
5. Celebrate effective usecases widely
6. Spoke 1:1 with non-adopters
7. Two months later, every engineer used every day

What were the reasons for non-adoption?

1. Uncertainty about setup
2. Low visibility into peer workflows
3. Non-standard tools not in usage dashboards
4. Poor initial experience
5. One morally opposed individual

Was this even a migration at all?

1. Migration of practices
2. Nuances drive solution
3. Avoiding “pressure without a plan”

(4) Ex 3: Adoption of remote harnesses

Local harnesses are great, but not enough:

1. Work stops when laptops turn off
2. High concurrency is messy
3. Can’t live dangerously
4. Reactive

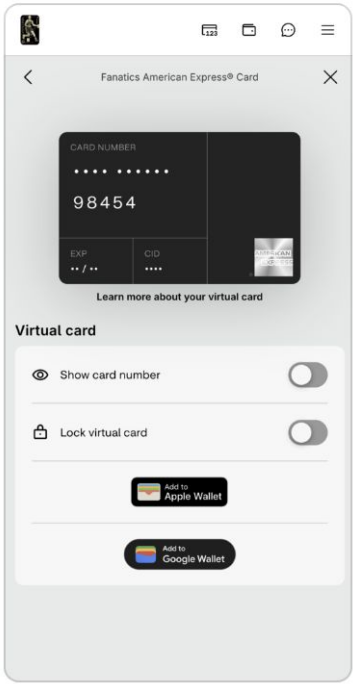
We also were running into

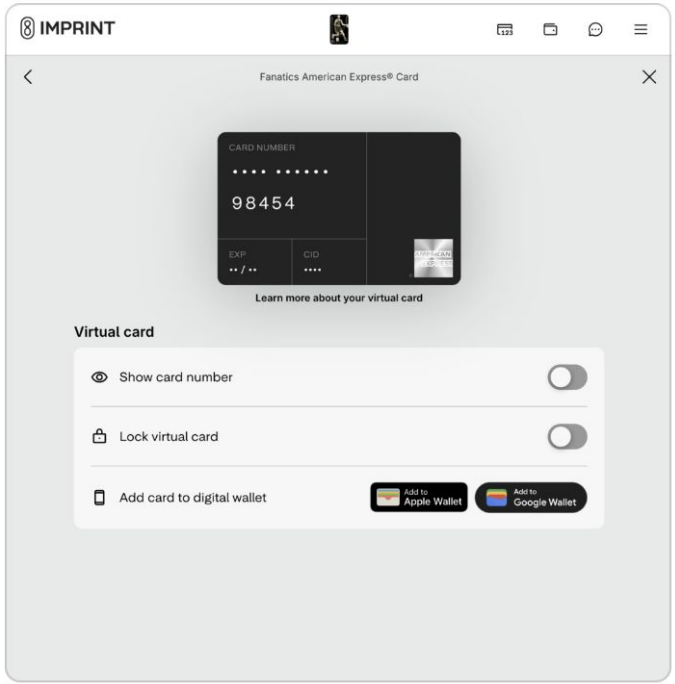
1. MCP friction
2. Fragment issue management
    a. Varied permissions
    b. Slack connectivity

Will Larson Thursday at 12:17 PM

@Linear file a ticket for MEX to display add to ios/add to google on same row (just without header) on mobile viewport on web, same as desktop (just omitting header on mobile vs tablet/desktop viewport). Delegate to AI Fleet

2 files ▾ | Download all





✅ 1

Two migrations:

1. Implementing and adopting Agent Fleet
2. Moving from JIRA to Linear

Migrating from JIRA to Linear:

1. Prototyped with Linear
2. Convinced JIRA power users to trial
3. Migration tooling (MCP to MCP)
4. #linear
5. A week in, decided to expand to all company
6. Everyone moved in <30 days
    a. all but ~7 spaces archived

Why this migration worked:

1. Executive led
2. Entire company, not just Tech team
3. Discovery
4. Culture of change
5. Ambitious early adopters
    a. Custom triaging harness
6. Relentless follow up with slow adopters :-)

(5) How we design migrations

My 2018 framework for migrations:

1. **Derisk** migration's usability by trying easiest case, and completeness by trying the hardest case
2. **Enable** migration by creating the (often self-service) tooling to complete it quickly
3. **Finish** the migration via strangler pattern, the tooling you've built, and doing it yourself
4. ~~**Delegate** the remaining steps to other teams~~ ← **Nope.**

How we've changed in 2026:

1. The core approach remains the same
2. Understanding deeply before solving remains elite. Avoid the "pressure without a plan" pattern
3. Almost any degree of migration complexity is preferable to cross-team dependencies and alignment
4. Already a terrible tool, delegation is an even worse tool for running migrations
5. "Migration taste" matters even more, because it's possible to start migrations with a single prompt

So these are just some basic migrations, huh?

1. Yup, that's basically right
2. However, each took at most 3 engineers and 2 months, despite being very large and impactful
3. It's possible to move very quickly at quality
4. And… we only have 1-2 years to catch up

Thank you!