SREDay 2026

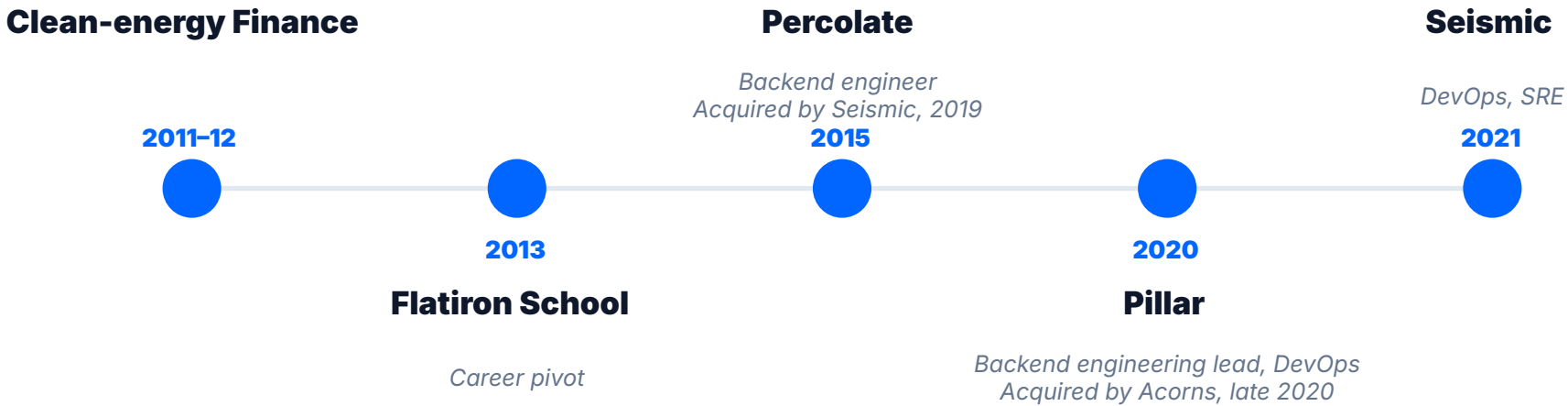
Shifting Left

Lessons Learned from 12 Years of Major SDLC Trends

Ian Miller · SRE, Seismic · ltce.dev

About Me

From clean-energy finance to SRE



Agenda

01 What does “Shifting Left” mean?

02 The four trends

- Monoliths → Microservices
- Site Reliability Engineering
- Kubernetes
- AI-assisted SDLC

03 What are the key takeaways?

THE THESIS

Shifting left isn't just about moving responsibilities earlier in the software development lifecycle.

It's about moving **ownership, capability, and accountability** closer to the people best positioned to act on them.

Architecture follows the org chart

Two drivers

01 Performance scaling — isolate hot domains so they scale independently.

02 Organizational scaling — teams want autonomy over their domain.

What shifted left:

Compute scaling, unit testing, and business-logic optimization moved out of central teams and into the domain team.

MONOLITH

one codebase · one team · one deploy

MICROSERVICES



each owned by a team · independent deploys

Sometimes the answer is in the middle

90%

infrastructure cost reduction

Amazon Prime Video, 2023 — moved a video monitoring workload from microservices + Step Functions back to a monolith on EC2.

THE HONEST READ

- Microservices is a scaling tool, not an architectural panacea.
- Coordination overhead is real and compounds quickly.
- The right shape matches your team count, business logic, and workload. It also can change over time.
- Organizations are fluid, and so are their problem spaces they solve. Service architecture should be too.

Kubernetes won. Now what?

82%

of container users run K8s in production

CNCF Annual Survey 2026 · up from 66% in 2023

Resource management

requests/limits, HPA, PDBs

Operational efficiency

Metrics API + autoscaling

Engineering productivity

ArgoCD, Helm, GitOps

Complexity didn't vanish - it shifted

“Developers are responsible for deploying their services to Kubernetes!”

— without Helm charts, CI/CD guardrails, observability, or access to expertise.

75%

of orgs cite the skills gap as their primary K8s blocker.

Red Hat State of K8s Security 2024

7.4

tools the average team juggles. 75% lose 6–15 hrs/week.

Port, State of IDPs 2025

YAML

the new programming language nobody asked to learn.

SRE: shifting reliability left, three ways

01 Teams own their SLOs

The team that builds defines reliable. Error budgets are the language for velocity vs reliability.

02 On-call lives with the code

The engineer who shipped is paged when it breaks. On-call pain creates feedback loops.

03 Self-service observability

SRE builds the platform; teams instrument themselves.

Centralized gatekeeping → distributed ownership, enabled by shared platforms.

Democratized observability, persistent gap

>50%

of organizations still report insufficient observability

Catchpoint SRE Report 2025

WHAT SELF-SERVICE DIDN'T FIX

- Instrumentation quality varies by team.
- Dashboards proliferate; signal-to-noise drops.
- Every team picks alert thresholds slightly differently.
- Knowing what to instrument is still a skill.

Novelty to default in 24 months

90%

of technologists use AI daily at work

DORA 2025 · +14 pts YoY · median 2 hrs/day

KEY FINDING

"AI doesn't fix a team. It amplifies what's already there."

Strong teams get stronger. Struggling teams ship dysfunction faster.

Individual gains ≠ org gains

INDIVIDUAL

+98%

more PRs merged per developer with AI assistance.

ORGANIZATIONAL

≈ flat

delivery throughput at the org level. Gains evaporate downstream.

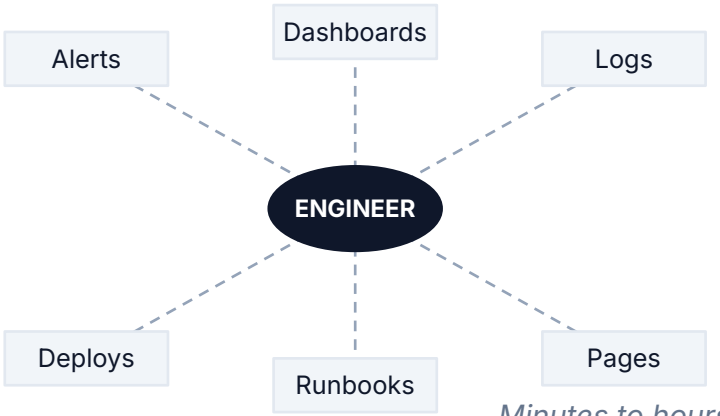
Faros AI telemetry, 10,000+ devs, 2025 · consistent with DORA 2025

More code generated faster, pushed into fragile pipelines = more incidents, not fewer.

Triage, before and after MCP

BEFORE

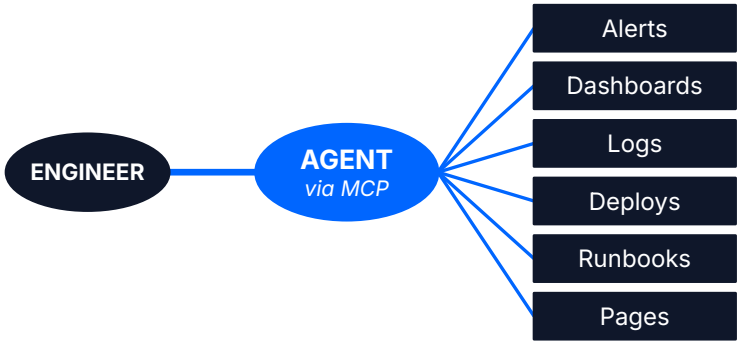
Engineer as integration layer



Minutes to hours.

AFTER

Agent as integration layer



Minutes — sometimes less.

Six tool tabs collapse into one conversation. The agent absorbs the fan-out… sometimes.

TAKEAWAYS

Shift left, or shove left?

THE CRITIQUE

- Devs now need to be experts in app code, tests, security, K8s, observability, FinOps, and compliance.
- Cognitive load is unbounded — and correlates negatively with every performance metric.
- Without tooling, training, and time, shift-left becomes shove-left.

THE RECEIPTS

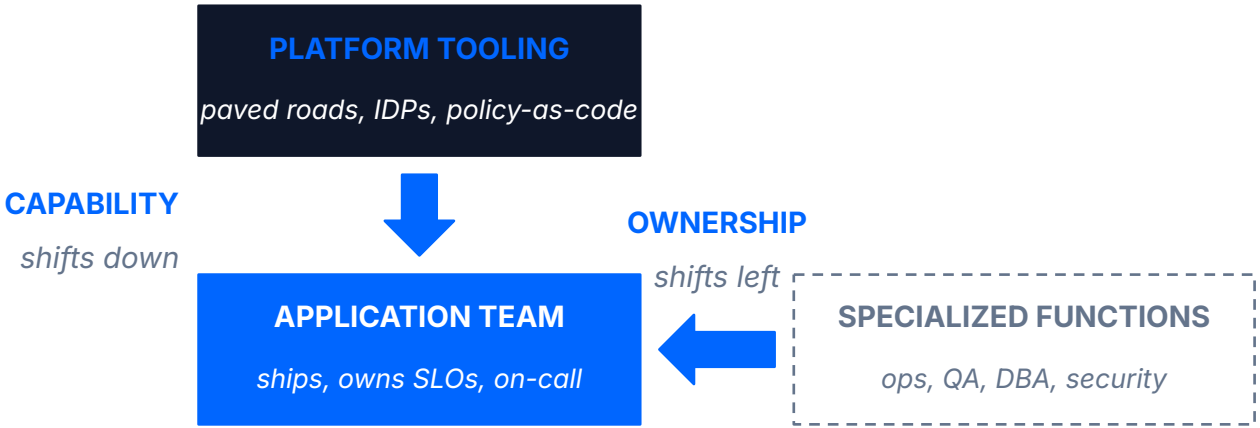
"Unbounded cognitive load has a negative impact on all performance indicators."

Puppet State of DevOps

75% of devs lose 6–15 hrs/week to tool sprawl.

Port, State of IDPs 2025

Platform Capabilities Enable Shifting Left



Capability and ownership converge on the team that ships.

Insights from the 4 Trends

01 **Each shift redistributed complexity. It didn't eliminate it.**

Microservices, K8s, SRE, AI — every win came with a new bill to pay somewhere else.

02 **Shift left works when capability shifts down to support it.**

Platforms aren't the opposite of shift-left. They're how the shift-left paradigm survives at scale.

03 **AI is an amplifier, not an answer.**

It magnifies your platform's strengths and weaknesses with equal enthusiasm. Fix the platform first.

Q&A

Ian Miller · SRE, Seismic

Original post available at ltce.dev/p/shifting-left