



STRATEGIC LEADERSHIP BRIEFING

The Self-Optimizing Infrastructure

How AI embedded across the entire SDLC keeps a **200K+ QPS** connected-device fleet fast, reliable, and profitable — at national scale.

When downtime means an unlocked front door

For a connected home-security fleet, reliability isn't a number on a dashboard — it's a promise. A missed doorbell notification or an eight-second live-view spin is a trust-destroying moment.

Tens of millions of connected devices

Millions of customers

Nationwide, every U.S. time zone



Failure is asymmetric

A dropped analytics event is a rounding error. A missed doorbell alert is a broken promise. **Not all errors cost the same** — so infra can't treat them the same.

If you only measure CPU, memory, and 5xx rates, you report **green** while customers experience **red**. Closing that gap is the whole game.

— THE SCALE

200,000+ requests per second — and the spikes are correlated

200K+

requests / second, steady state

10K+

servers to absorb the load

<4s

notification latency budget, E2E

2–3x

burst multiple in minutes

Bursty + correlated: holidays, deliveries, and regional weather drive millions of synchronized events. Averages hide the moments that matter.

• QPS, server count, and burst multiple are **illustrative at fleet scale**. The <4s latency SLO and 10M+ daily-event volume are from published operational data.



Requests per second over 24h

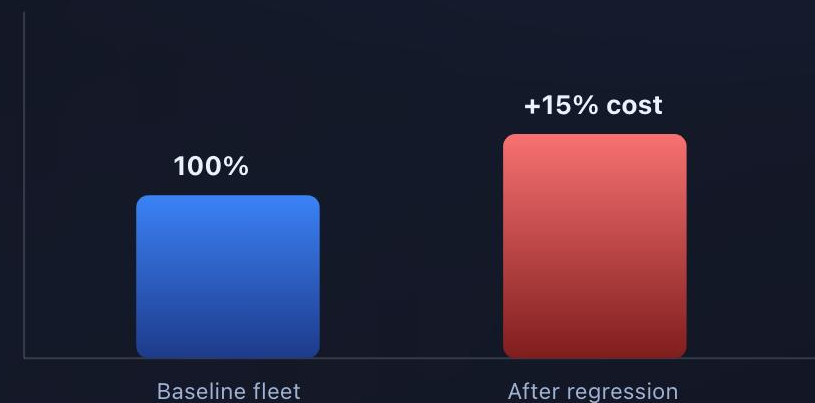


In this domain, **API performance is the P&L**

- \$ Compute is the largest line item.** Tens of thousands of servers means efficiency is a balance-sheet decision, not a micro-optimization.
- % Regressions multiply by fleet size.** A 15% CPU regression on a hot path = ~15% more servers to hold the same p99.
- ⚡ The market is unforgiving.** Competitors ship weekly. Slow or expensive both lose — you must be both fast and lean to stay profitable.



Cost of a 15% hot-path regression



Recurring, every month, on your biggest cost center — before the customer ever notices the latency.

Operational data has outpaced human analysis

Across tens of thousands of servers, every release emits payload changes, CPU/memory/IO signatures, latency distributions, and error signals. No human team can correlate that in the minutes an error budget gives you.

The traditional loop — notice → investigate → correlate → patch → verify — is too slow when a regression burns budget in real time.



The signal-to-human gap

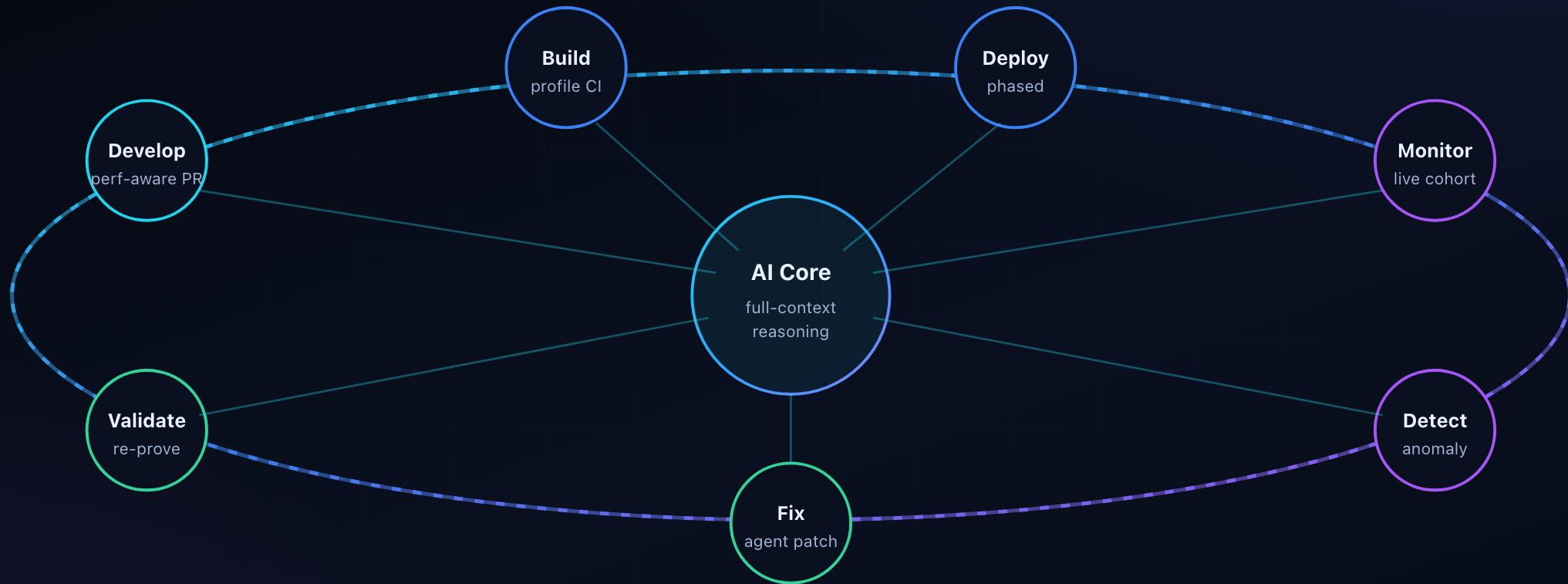


The thesis

AI belongs **inside** the SDLC — not on a dashboard beside it.

From development to deployment to release monitoring, an AI layer **observes → detects → fixes → validates** — a self-driving loop that optimizes infrastructure efficiency without waiting for a human to notice.

A self-driving loop across the entire lifecycle



Shift performance **left** — before code merges



Perf-aware review

AI reviews every PR for efficiency, not just correctness — hot-path allocations, N+1 queries, unbounded payloads.



Predicted impact

Each change is annotated with likely CPU / memory / IO effect at fleet scale, in the PR itself.



Cheapest place to fix

A regression caught here costs a comment. In production it costs an incident **plus** a fleet-sized capacity hit.

"Shift-left" applied to **cost**, not just defects — the first checkpoint in the loop.

Validate the **payload** → **resource** signature

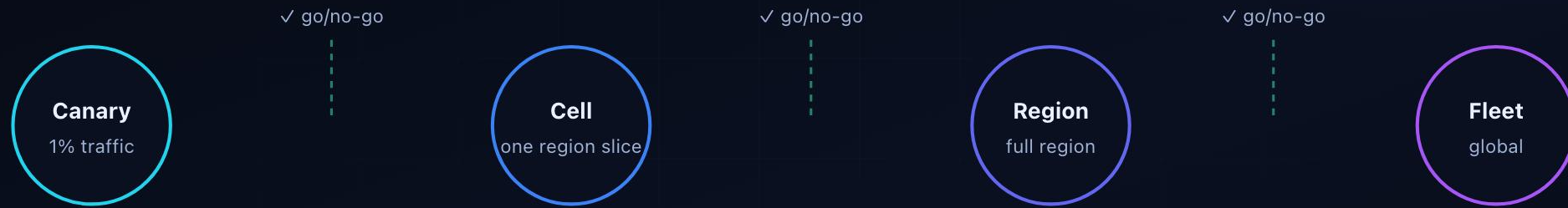
- 1 **Replay real-shaped payloads** against the new build in CI — including peak-burst shapes, not just averages.
- 2 **Profile CPU / memory / IO + tail latency** and fingerprint the resource signature of the change.
- 3 **Diff vs known-good baseline.** Output is a verdict — "p99 CPU +12% on this function" — not a wall of graphs.
- 4 **Auto-hold on budget breach.** The regression gate has authority to stop the release.



Resource signature: new build vs baseline



Contain blast radius — **canary** → **cell** → **region**



AI monitors each cohort vs. fleet control group:
p99 latency · error rate · CPU / memory / IO signature

Catch the **quiet** regression — and name its cause



p99 latency vs. learned normal band



Learn "normal," don't hard-code it. Baselines beat static thresholds for silent regressions.

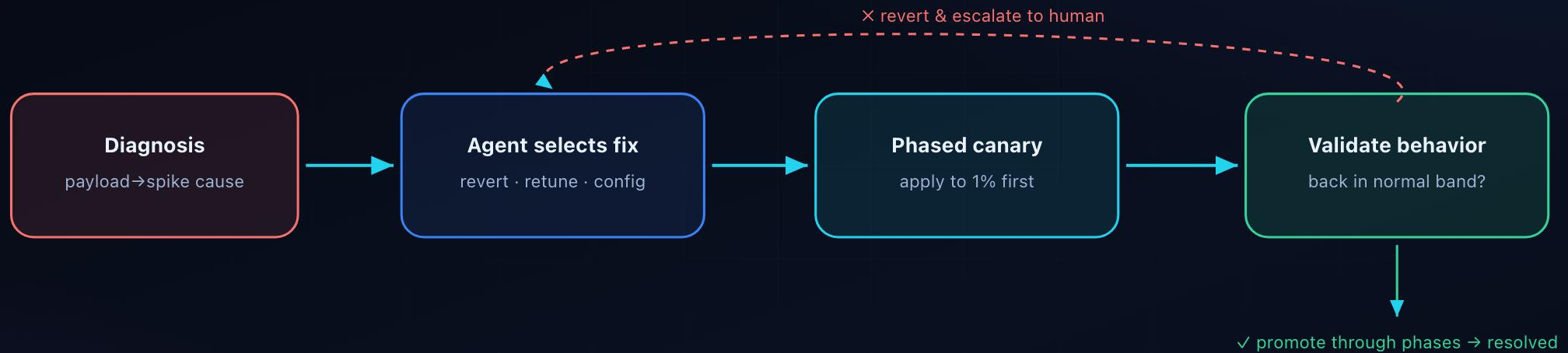


Correlate cause to effect. Tie the payload / code change to the exact CPU / memory / IO spike.



Wired to action. Detection triggers a hold and hands off to remediation — no human-in-the-middle delay.

A second agent **fixes it** — then **proves it**



Every autonomous action is **blast-radius-contained, logged, and reversible**. The machine owns the speed; the human owns the high-stakes call.

The result

Infrastructure that **optimizes itself** — continuously, within safe bounds

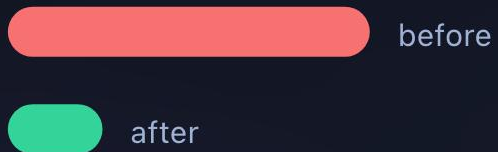


One AI core, full context, driving every stage — reclaiming capacity, protecting p99, and defending margin without a human in the critical path.

Faster, leaner, more reliable — **compounding**

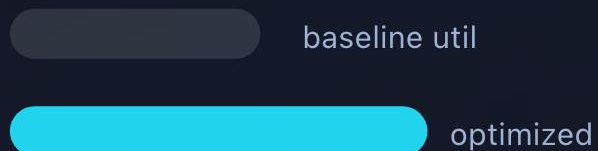
↓ MTTR

Detection wired to remediation collapses time-to-mitigate from human-paced hours to machine-paced minutes.



↑ Efficiency

Regressions stopped pre-fleet + continuous right-sizing reclaim capacity on your biggest cost center.



↓ Incidents

Phased validation + blast-radius control means a bad change costs a cell, not a country.



contained to one cell →

On tens of thousands of servers, single-digit-percent efficiency = balance-sheet-scale savings, recurring every month.

From AI-as-dashboard to AI-as-operator

- **Make the loop the standard.** Self-optimizing SDLC as a first-class design principle, not a bolt-on.
- **Grant bounded authority.** Let AI act within safe, reversible, blast-radius-contained limits.
- **Design for the AI era up front.** Predictive capacity, native zero-trust, self-healing efficiency.

Stay fast · stay lean · stay profitable — while the fleet keeps growing



The one-line takeaway

At 200K+ QPS, the winning infrastructure is the one that optimizes itself.

Thank you

Let's build infrastructure that runs itself.

Questions welcome.