The Pipes Are Finally Moving: Why Clinical Event Streaming Is the Infrastructure Bet Nobody Took Seriously Enough

MAR 08, 2026 • PAID

Abstract

This essay covers the architectural shift from batch ETL to event-driven clinical pipelines, with specific focus on EHR event streaming, real-time deterioration detection, and operational alert routing. Topics include why healthcare streamin categorically harder than fintech equivalents, what the dominant infrastructure looks like, and where the actual venture opportunity sits.

Key Points:

- Batch ETL has been healthcare’s dominant data pattern for 30+ years and it wa always wrong for clinical work

- Event-driven architectures using Kafka, Flink, and similar tooling are now viab hospital environments

- Clinical data has 5-6 structural problems fintech doesn’t have: schema chaos, contextual validity, documentation timing lag, regulatory constraints, human-in- requirements, and no equivalent to atomic transactions

- Real-time deterioration detection (sepsis, respiratory failure) is the clearest cli proof case, with measurable mortality impact

- The venture opportunity is not the streaming infrastructure itself but the clini logic layer sitting on top of it

- Health systems have the data and the willingness; they largely lack the enginee talent to build this internally

Table of Contents:

The Batch ETL Hangover

The honest story of healthcare data infrastructure is that it was designed around billing cycle, not the patient. Everything in the legacy stack optimizes for the cla Data gets captured, batched, transformed, and exported in windows that align w when someone needs to submit something to a payer or generate a report for the board. The clinical workflow was an afterthought, and the infrastructure reflecte that priority ordering pretty faithfully for about three decades.

Batch ETL as a pattern made sense for a certain era. You pull data from an EHR 2am, transform it, load it into a warehouse, and analysts run reports in the morn That is genuinely fine for retrospective quality analysis, population health dashboards, and financial reconciliation. Nobody needs real-time data to figure o how the readmission rate trended in Q3. The problem is that a lot of healthcare

cases are not retrospective. A patient who is septic at 11pm cannot wait for the 2 batch. A nurse who needs to know a patient's current medication list before administering something cannot work from yesterday's extract. The infrastructu was fundamentally mismatched with the urgency of clinical reality, and the indu just kind of tolerated that mismatch for a long time because building something better was genuinely hard and the incumbents had no financial incentive to do it

What changed is a combination of three things happening more or less simultaneously. Epic and the other major EHR vendors started building real API surfaces, particularly after CMS started mandating interoperability. Cloud infrastructure got cheap enough that health systems could actually consider run Kafka clusters without having to justify a seven-figure capex line item. And a generation of engineers who had built streaming systems in fintech and adtech s showing up in healthcare companies with a reasonable question: why is this so m worse than what we built at Stripe? Those three forces together created the conc for the current architectural transition.

The transition is real and it is happening, but it is not happening uniformly. Wh see in the market right now is a fairly wide spectrum ranging from academic me centers with genuinely sophisticated real-time pipelines down to rural critical a hospitals still faxing things. The opportunity space lives in that gap, and understanding the architecture is table stakes for understanding where the real leverage points are.

What Event-Driven Actually Means in a Hospital Context

Event-driven architecture in a generic software context means building systems around events rather than state. Instead of periodically polling a database to see something changed, you emit an event when the change happens, and downstrea consumers react to that event in real time or near-real time. The architectural pa enables loose coupling, horizontal scalability, and latency profiles that batch processing structurally cannot achieve.

In a hospital, an event is something like: a new lab result has been resulted, a medication has been administered, a vital sign has been recorded, a patient has b admitted to the ICU, a nursing note has been signed. These are all discrete clini events that happen at a specific time and carry specific clinical meaning. In a bat world, these events get written to database tables and sit there until something queries them. In an event-driven world, the occurrence of the event itself trigge downstream processing in real time.

The EHR is the source of truth for most clinical events, and the shift toward FH based APIs has made EHR event streams meaningfully more accessible than they even five years ago. Epic's FHIR R4 implementation, Cerner's (now Oracle Healt equivalent, and the smaller vendors to varying degrees all support subscription models where external systems can register to receive notifications when specific clinical events occur. This is the foundation of the clinical event streaming patte using the EHR as an event source, extracting those events through a standards-b API layer, and routing them into a streaming infrastructure for downstream processing.

The practical architecture involves several layers. The EHR emits events through API surface. Those events get ingested into a message broker, with Kafka being dominant choice for anything at serious scale. Consumers subscribe to relevant streams and process those events in real time, applying clinical logic to generate alerts, update downstream systems, or trigger workflows. The output might be a to a nurse, an update to a care coordination platform, a flag in a clinical decision support tool, or an entry in a real-time analytics system. The pattern is not complicated at the conceptual level. The implementation is where it gets very ha

The Stack: Kafka, Flink, and the Middleware Nobody Talks About

Apache Kafka is the backbone of most serious clinical event streaming implementations. Kafka is a distributed event streaming platform originally buil LinkedIn that has become the de facto standard for high-throughput, fault-toler

message streaming across industries. In a clinical context, Kafka functions as the central nervous system: events from the EHR and other clinical systems get publ to Kafka topics, consumers read from those topics, and the log-based architectu means events are durable and replayable, which matters enormously for audit an compliance reasons.

Kafka's design properties are well-suited to healthcare. The immutable log mean have a complete audit trail of every event in sequence. Partition-based scaling m you can handle high-volume event streams without architectural rethinking. Consumer group semantics mean you can have multiple downstream systems independently consuming the same event stream without coordination overhead Retention policies mean you can replay historical events for model training, debugging, or compliance review. None of this is magic; it is just good distribute systems design, and it translates well to clinical environments where auditability reliability are non-negotiable.

Apache Flink is the stream processing layer that most serious implementations u top of Kafka. Where Kafka handles the transport and durability of events, Flink handles the processing logic: windowing, stateful computation, complex event processing, and real-time aggregations. A typical deterioration detection pipelin might ingest vital signs, labs, and nursing assessments from Kafka, use Flink to compute rolling aggregates and apply scoring logic in real time, and emit alerts to Kafka for downstream routing. The Flink ecosystem has matured significantly its support for exactly-once semantics is particularly relevant in healthcare wher really cannot afford to double-count a medication administration or miss a lab r

The middleware layer is where a lot of implementations quietly fall apart and wh the interesting engineering is happening. Between the EHR API surface and the ingestion layer, you need something that handles authentication, rate limiting, s validation, error handling, and the translation between whatever the EHR is emi and whatever your internal event schema looks like. In fintech, this middleware fairly well-standardized around things like Plaid for data access and established protocol implementations for market data. In healthcare, this layer is largely bes and the combination of EHR-specific API behaviors, HL7 FHIR's considerable

flexibility (which is really a polite way of saying inconsistency), and health syster security requirements makes this layer disproportionately expensive to build an maintain.

Several companies have emerged trying to standardize this integration layer, essentially providing a managed event streaming infrastructure for healthcare th abstracts the EHR connectivity problem. The pitch is straightforward: instead o building your own EHR integration team and maintaining connectors for Epic, Cerner, Meditech, and twelve others, you subscribe to a service that delivers a cl normalized event stream. The value proposition is real, but the market structure tricky because health systems have strong incentives to own this layer themselve they become more sophisticated, and EHR vendors have obvious incentives to ga

Why This Is Not Fintech (And Fintech People Keep Getting Burned)

This is the section that matters most for people coming into healthcare from fin or general enterprise software backgrounds, because the pattern-matching is seductive and wrong in ways that cost a lot of time and money.

The surface-level similarity is real. Fintech built real-time streaming infrastruct for fraud detection, payments processing, and market data consumption years be healthcare got serious about it. The tooling is largely the same: Kafka, Flink, sim cloud infrastructure. The business case rhymes: low-latency detection of bad outcomes, real-time routing of actionable information. Engineers who built frau detection pipelines at a payments company look at sepsis detection pipelines an the same problem. They are not crazy; there is genuine structural similarity. But are five or six deep differences that make healthcare streaming categorically har and ignoring any one of them will hurt you.

The first is schema chaos. Financial transactions have extremely well-defined scl An ACH transfer has a sender, receiver, amount, timestamp, and a routing numb The schema is standardized, enforced at the protocol level, and has been stable f decades. Clinical data is the opposite. A blood pressure measurement might be

recorded as a discrete vital sign, embedded in a nursing note, captured in a devi integration feed, documented as part of a procedure note, or exist in all four pla with subtly different values because of calibration differences or documentation timing. FHIR is supposed to solve this, but FHIR's flexibility means that the sam clinical concept can be validly represented in dozens of different ways dependin the EHR version, the health system's configuration, and the individual clinician's documentation habits. Building a streaming pipeline that correctly normalizes t requires clinical domain expertise that most engineering teams simply do not ha

The second is contextual validity, which has no real fintech analogue. A lab resu 1.2 for serum creatinine means something very different depending on the patie age, baseline kidney function, recent fluid status, and whether they are on nephr medications. Financial data is largely self-contained: a transaction either happer it did not, the amount is what it is, and fraud signals are statistical properties of transaction itself. Clinical event streams require downstream logic that is deeply contextual, which means your streaming pipeline needs access to a rich patient context to interpret any given event correctly. This is a state management proble with very high dimensionality, and it does not simplify cleanly.

The third is documentation timing lag, which breaks a lot of naive real-time assumptions. In fintech, the event and the documentation of the event are essent simultaneous. The transaction either posts or it does not. In a clinical environme there can be hours between when something clinically significant happens and w it gets documented in the EHR. A nurse might administer a medication at 2pm a document it at 4pm when the patient stabilizes enough to let her chart. A physic might update a problem list during a late-night catch-up session reflecting assessments made during rounds that morning. Your event stream reflects documentation time, not clinical time, and the gap between the two is not rando noise; it is systematically correlated with clinical busyness, which is itself correl with patient acuity. Any model trained on event streams without accounting for will have systematic bias in exactly the situations where you need it to work best

The fourth is the regulatory and liability architecture. Financial transactions ha compliance requirements that are serious but largely well-defined: BSA/AML,

transaction monitoring, record retention. The regulators are sophisticated about technology. In healthcare, the regulatory surface is broader, more ambiguous, an carries direct patient safety liability. HIPAA's requirements around PHI handlin streaming pipelines add complexity around encryption, access logging, and brea notification that fintech compliance teams do not deal with. More importantly, i clinical decision support alert fires incorrectly and a clinician acts on it and a pa is harmed, the liability exposure is categorically different from a fraud model tha incorrectly flags a legitimate transaction. This changes the risk tolerance for shi imperfect systems, which changes the development velocity, which changes the business model math.

The fifth is the human-in-loop requirement. Fintech automation is largely truste act autonomously on streaming decisions within defined parameters. A fraud detection system can freeze a card without human review because the cost of a fa positive (inconvenience) is vastly lower than the cost of a false negative (fraud los Clinical alerting is asymmetric in a different way. You need a human to make the clinical decision, and that human is busy, cognitively loaded, and will start ignor alerts if they are not highly precise. Alert fatigue is a documented clinical proble with measurable patient safety consequences. Sepsis alerts that fire too broadly g tuned out, and then the real sepsis cases get missed. This means your streaming cannot optimize purely for recall; it has to achieve a precision-recall balance tha keeps alert rates at levels clinicians will actually respond to, which is a much har optimization target than fintech fraud thresholds.

The sixth, which does not get discussed enough, is the absence of atomic transac Fintech data integrity is anchored to transactional semantics: money either move it did not, and the database record reflects ground truth. Clinical data does not h an equivalent. A patient's true clinical state is not a database record; it is a physi reality that the documentation system imperfectly captures. Two nurses can doc conflicting vital signs taken five minutes apart and both be correct within measurement error. A medication can be ordered, canceled, re-ordered, adminis and then have the administration retracted due to a documentation error, all wit few hours, creating an event stream that is genuinely ambiguous about what actu

happened. Building streaming logic that handles this gracefully requires both technical sophistication and clinical domain knowledge working together.

Deterioration Detection as the Canonic Use Case

Sepsis is the right starting point for any concrete discussion of clinical event streaming because it combines clinical urgency, measurable outcome data, reaso streaming tractability, and a large installed base of attempted implementations a health systems. It is also a case study in what separates good streaming architect from bad ones.

Sepsis kills somewhere around 270,000 Americans per year and accounts for rou one in three hospital deaths, making it the leading cause of hospital mortality. T clinical management principle is simple and well-established: early identificatio treatment dramatically improves outcomes. The three-hour bundle, which inclu blood cultures, lactate measurement, and antibiotic administration, has strong evidence behind it. The problem has always been early identification. Sepsis presentation is heterogeneous, its early signs overlap with many non-septic conditions, and the clinical progression can be fast enough that a patient who lo borderline at 6pm is in septic shock by 10pm.

The streaming architecture for deterioration detection typically works somethin this. Vital signs from bedside monitoring devices and nursing documentation fe into the event stream in near real time. Lab results flow in as they are resulted ir laboratory information system. Nursing assessments, medication administration intake/output documentation contribute additional event streams. A stateful str processor, usually something like Flink or Spark Streaming, maintains a running patient state object and applies scoring logic in real time as new events arrive. T scoring might be rule-based (variations on SIRS criteria or qSOFA), model-based classifiers trained on historical sepsis cases), or hybrid. When the score exceeds a threshold, an alert fires into the clinical workflow.

The implementations that work have several things in common. They have tight feedback loops with clinical champions who help tune the alert thresholds and understand why specific false positives are occurring. They invest heavily in the quality layer, particularly around vital sign integration from bedside devices, wh turns out to be surprisingly messy because device integration is its own can of w They treat the alert routing as a first-class problem, meaning the question of wh paged, through what channel, with what information, and with what escalation l no one responds gets as much engineering attention as the detection logic itself. they measure outcome metrics, not just alert volume, so they can actually assess whether the system is improving mortality rather than just generating activity.

The implementations that fail typically make one of a few predictable mistakes. build the detection logic before solving the data quality problem, which means t are running sophisticated algorithms on garbage inputs. They set alert threshold without clinical input and end up with either alert fatigue from over-firing or m cases from under-firing. They treat the project as a one-time build rather than a ongoing system requiring continuous calibration as patient populations and care patterns change. Or they build in isolation from the EHR workflow, creating a p alert system that clinicians learn to ignore because it is not integrated into wher actually work.

Beyond sepsis, the deterioration detection pattern generalizes to respiratory fail acute kidney injury, hemodynamic instability, and fall risk, all of which share th structural characteristics that make streaming tractable: there are leading indica that show up in the data stream before the clinical event, there is a meaningful intervention window, and the outcome is measurable. The market for clinical deterioration solutions is real, with companies like Philips, GE HealthCare, Mas and a generation of digital health startups all competing for health system contr The differentiation increasingly lives in the algorithm quality and the workflow integration, not the streaming infrastructure itself.

Streaming Into Operations: The Alert Routing Problem

Building a streaming pipeline that detects a clinical event in real time is necessa not sufficient. The other half of the problem, which gets proportionally less atte in technical discussions, is routing that detection into the operational workflow way that actually changes what happens to the patient. This is the alert routing problem, and it is where a lot of technically excellent implementations have quie failed to generate clinical impact.

The operational context matters enormously. A hospital at 3am with fully staffed has very different routing constraints than the same hospital on a Friday afterno when the day team is handing off and the night team is just coming on. The righ recipient for a sepsis alert might be the bedside nurse, the rapid response team, attending physician, or a charge nurse depending on the patient's location, the severity of the alert, the current staffing configuration, and whether previous ale have already been acknowledged. None of this routing logic is static, and buildir in a way that reflects actual hospital operations rather than the org chart on a whiteboard requires deep operational knowledge.

The technical architecture for alert routing typically involves a separate set of microservices sitting downstream of the detection layer, consuming detected eve from Kafka and applying routing logic to determine who gets notified, through v channel, and with what clinical context attached. The notification channels themselves are a small integration project: most hospitals have some combinatio paging systems, secure messaging apps (Epic's In Basket, Vocera, TigerConnect a common), EHR-native alert mechanisms, and in some cases direct integration wi nurse call systems. Building connectors to all of these and maintaining them as vendors update their APIs is meaningful ongoing engineering work.

The escalation logic is where it gets interesting from a systems design perspectiv sepsis alert fires and is not acknowledged within ten minutes, what happens? Yo need a defined escalation path, and that escalation path needs to be configurable unit, per time of day, and per alert severity. You also need feedback loops: when clinician acknowledges and dismisses an alert, that dismissal should feed back ir the system both for audit purposes and potentially for algorithm calibration. Wh clinician acknowledges and takes action, the downstream clinical events (orderir

blood cultures, starting antibiotics) should close the alert loop and update syste state. This event-driven feedback architecture means your operational alerting s is itself producing events that feed back into the clinical event stream, creating a closed loop that is genuinely complex to reason about but creates real opportuni for continuous improvement.

The integration with EHR-native workflows is the crux of adoption. Clinicians d want to check a separate application to see alerts; they want alerts to surface in t workflow they are already in. Epic's CDS Hooks implementation provides a stan based mechanism for injecting clinical decision support alerts into the physician workflow at defined trigger points, and building on top of that standard rather t creating a parallel interface dramatically improves adoption rates. The same prin applies to nursing workflows: alerts that surface in the nurse's existing documen workflow get acted on; alerts that require opening a different application mostly ignored.

Where the Venture Opportunity Actual Lives

The streaming infrastructure layer is largely commoditized or rapidly commodit Running Kafka on cloud infrastructure is not a competitive differentiator; it is t stakes. The interesting venture territory is not the pipes themselves but what flo through them, who controls the clinical logic, and who owns the integration laye

The clinical logic layer is the first place worth examining carefully. The value in deterioration detection is not the ability to run Flink; it is having a sepsis algori that outperforms Epic's built-in DETERIORATION INDEX, having the training to prove it, and having the clinical validation studies to sell against it. The same principle applies across any streaming-powered clinical application. The sustain competitive advantage is the clinical IP, the data network effects from deployme across multiple health systems, and the relationships with clinical champions wh speak to outcomes. This is a hard moat to build but a real one, and it is why the

clinical decision support companies that have gotten this right command meani
revenue multiples.

The EHR integration layer is the second high-value territory. Getting clean, normalized, real-time event streams out of EHR systems is genuinely hard, and h systems lack the internal engineering capacity to do it well at scale. Companies t have solved this problem across multiple EHR vendors and multiple health syste configurations have infrastructure that is expensive to replicate and creates real switching costs. The business model challenge is that EHR vendors, particularly have strong incentives to own this layer and have been steadily expanding their c connectivity products. Playing in this space requires a clear theory on why third integration infrastructure wins over the long term against an EHR vendor that controls the underlying data.

The operational workflow integration layer is probably the most underappreciat opportunity. The alert routing problem described above is hard enough that mos health systems solve it poorly, and the gap between a well-designed alert routing system and a poorly designed one has measurable clinical impact. Companies tha build deep operational workflow integration, meaning they understand staffing patterns, care team structures, escalation protocols, and communication prefere at the unit level, are solving a problem that requires both software engineering a operational expertise that is genuinely scarce. The sales motion is harder becaus are selling operational change, not just software, but the retention dynamics are excellent because these systems become deeply embedded in how a unit actually functions.

The health system analytics layer is where a lot of the near-term revenue is, even is not the highest-value long-term position. Health systems want to understand v is happening in their patient population in real time, not just retrospectively. Re time operational dashboards, bed management tools, staffing optimization syste and length-of-stay prediction tools all benefit from event-driven data infrastruct The buyers are typically COOs and CNOs rather than CIOs, and the purchase is justified on operational efficiency rather than clinical outcomes, which makes th sales cycle shorter and the ROI story cleaner. The risk is that these applications

relatively easy to replicate once the underlying streaming infrastructure exists, v
limits pricing power unless you are also providing the infrastructure layer.

The interesting positioning question for founders in this space is whether to go
narrow and deep (own one clinical domain, like sepsis or AKI, and become the
definitive solution for that problem across every health system) or broad and pla
(build the infrastructure and the clinical logic layer and the workflow integratio
try to cover multiple use cases). The narrow plays have cleaner product focus an
faster time to clinical validation but face a natural ceiling. The platform plays re
more capital, longer development cycles, and the organizational discipline to no
distracted, but create larger outcome potential. Both approaches have worked; w
has not worked is being vague about which you are doing.

The talent question is also worth being direct about. Building this stuff well req
clinical informaticists who understand how EHR data is structured and what it r
distributed systems engineers who have built high-throughput streaming pipeli
and people who understand both domains well enough to translate between ther
That combination is genuinely rare. Health systems have the clinical expertise b
rarely the distributed systems talent. Fintech shops have the distributed systems
talent but rarely the clinical domain expertise. The companies that figure out th
talent equation are building something that is hard to reproduce even with capit

The macro tailwind is real: CMS interoperability mandates, TEFCA, the continu
expansion of Epic's and Oracle's API surfaces, and the increasing sophistication
health system IT teams all point in the direction of more accessible clinical even
over time. The bottleneck has shifted from data access to the harder problem of
something clinically meaningful with the data in real time. That is where the ne
generation of healthcare infrastructure companies is going to be built, and the
window for establishing durable positions in the clinical logic and workflow
integration layers is probably shorter than it looks from the outside.

3 Likes • 1 Restack

← Previous

Nex

Discussion about this post

Comments Restacks

Write a comment...