CLIXLOGIX CONSULTING PRACTICE

TECHNOLOGY STRATEGY AND ARCHITECTURE REPORT

Prepared for board or investor diligence

Prepared for

[Client Confidential]

Engagement

Platform Modernization and Target-State Architecture: Scaling from 3,000 to 10,000 Buildings Under Management

Date	Document Version	Classification	Prepared by
Q2 2025	v1.0	Strictly Confidential	Clixlogix Consulting Practice

Table of Contents

SECTION 1

Executive Summary

Recommendation: Modernize the Client's IoT-SaaS platform through an 18-month, three-phase program that separates the telemetry-processing path from the application request path, replaces synchronous PostgreSQL-based telemetry ingestion with an event-driven streaming architecture, and delivers multi-region active-active availability for the application tier. The program positions the Company to reach 10,000 buildings under management and $60M ARR without architectural constraint.

The Client's current platform is a coherent system for a $20M ARR business operating 3,000 buildings. It is not a coherent system for the Company they intend to be in 24 months. Three structural constraints limit growth: the IoT telemetry path shares compute and storage resources with the application request path, making both susceptible to load from the other; the notification fan-out runs synchronously in the application process, producing cascading failures under alert bursts; and the data layer carries both transactional operational data and high-volume time-series telemetry in a single PostgreSQL instance with no partitioning or retention strategy.

Supporting Rationale

- At 10,000 buildings and 1,000,000 connected devices, the IoT ingestion path will receive an estimated peak load of 8,500 telemetry messages per second. The current synchronous write-to-PostgreSQL path degrades at approximately 2,200 messages per second under the resource contention observed during Q4 2024 and Q1 2025 incident reviews.
- The notification fan-out has produced three production incidents in six months. Each incident resulted from synchronous HTTP calls to Twilio, SendGrid, or Firebase failing under burst load and propagating failures back into the alert processing pipeline.
- PostgreSQL is the wrong storage medium for 1.4 terabytes of time-series telemetry data generated annually at 10,000-building scale. Read performance on time-range queries degrades predictably as table size grows past the 500 gigabyte range without partitioning.
- Multi-tenant data isolation has been flagged in the Client's most recent security review. The current implementation relies on application-level tenancy enforcement without database-level row security.
- The AI anomaly detection capability the Company intends to commercialize requires a structured time-series data foundation. The current architecture has no pathway to that foundation without the modernization described in this report.

Key Trade-offs Accepted

The recommended architecture accepts higher operational complexity in exchange for the scale and reliability characteristics the business requires. A monolithic Rails application is operationally simpler than twelve domain services on Kubernetes. The business case for the complexity increase rests on two facts: the current architecture will fail at the target scale, and the twelve services map to workloads with sufficiently different performance and scaling profiles that operating them together would require over-provisioning the entire system to accommodate the highest-demand workload.

Projected Cost Envelope

Category	18-Month Build Cost	Year 1 Run (3K bldgs)	Year 2 Run (6K bldgs)	Year 3 Run (10K bldgs)
Engineering and program	$3.2M - $4.1M	N/A	N/A	N/A
Cloud infrastructure (AWS)	Included in build	$420K - $480K	$680K - $780K	$980K - $1.1M
Observability (Datadog)	Included in build	$120K - $150K	$180K - $220K	$240K - $290K
Security tooling	Included in build	$60K - $80K	$80K - $100K	$100K - $130K
Third-party services	Included in build	$110K - $140K	$160K - $200K	$210K - $260K
Total programme investment	$3.2M - $4.1M	$710K - $850K	$1.1M - $1.3M	$1.53M - $1.78M

Top Risks

- Data migration from PostgreSQL telemetry tables to TimescaleDB introduces dual-write complexity and potential consistency gaps. Mitigation: shadow-mode validation for a minimum of 30 days before the legacy path is decommissioned.
- The engineering team of 35, with no dedicated platform team, will execute production feature development in parallel with the architectural transformation. Mitigation: phase sequencing isolates transformation workstreams from product delivery.
- Series C diligence may accelerate the SOC 2 timeline. Mitigation: security baseline work begins in Phase 1, structured to produce audit artifacts progressively.

Timeline Summary

Phase 1 (months 0 to 6) establishes the foundation: streaming layer, time-series database, observability, and ingestion-path separation. Phase 2 (months 6 to 12) migrates the notification layer, prepares for multi-region, and completes the most critical service decompositions. Phase 3 (months 12 to 18) activates multi-region active-active, brings AI anomaly detection into production, and completes the hardware fleet management modernization.

SECTION 2

Methodology

This assessment was conducted over a six-week engagement spanning March and April 2025. Evidence was gathered across seven primary streams, each designed to test a specific hypothesis about the current architecture's constraints and the viability of the recommended target state.

Evidence Streams

Evidence Stream	Scope and Approach
Architecture review sessions	Four half-day working sessions with the CTO, VP Engineering, and two senior staff engineers covering application architecture, IoT data path, data layer design, and deployment model.
Code repository inspection	Review of the main Rails application repository, the notification service, and the IoT ingestion endpoint. Static analysis applied to identify N+1 patterns, missing indices, and synchronous HTTP call sites.
Infrastructure audit	Review of AWS account architecture, EC2 sizing, RDS configuration, Redis configuration, security group rules, IAM role structure, and networking topology.
Performance and observability baseline	Analysis of CloudWatch metrics, PostgreSQL slow query logs, and application performance data from October 2024 through March 2025.
IoT telemetry flow analysis	End-to-end tracing of the MQTT-to-PostgreSQL ingestion path including device connection behaviour, message arrival rates, batch sizes, and write amplification patterns.
Incident post-mortem review	Review of post-mortem documentation for the three notification service production incidents in Q4 2024 and Q1 2025.
Vendor and technology research	Assessment of streaming platforms, time-series databases, observability tooling, and IoT device management platforms against the Company's specific requirements.

Stakeholder Interviews

The following roles were interviewed. All participants are referenced by role only.

- CTO
- VP Engineering
- Head of Platform
- Two senior staff engineers (application platform and IoT path)

- Head of Security
- Head of Hardware Operations
- Two SREs
- VP Product (one session, AI capability requirements and product roadmap context)

Scope Exclusions

- Business model, pricing strategy, or go-to-market approach
- Hardware device design, embedded firmware source code review, or RF network design
- Procurement vendor evaluation or contract negotiation
- Organisational design or engineering team structure
- Detailed financial modelling for investor materials beyond the cost projections in Section 13

SECTION 3

Current State Assessment

The Client's current platform represents a competent implementation of a Rails-based multi-tenant SaaS, built by a focused engineering team that has prioritised feature delivery over infrastructure investment. At $20M ARR, 3,000 buildings, and 300,000 connected devices, the architectural debt is no longer deferred cost -- it is active constraint.

Application Architecture

The primary application is a Rails 6.1 monolith that handles IoT telemetry ingestion, user-facing HTTP requests, background job processing, and notification fan-out within a single application boundary. The IoT ingestion endpoint and the user-facing dashboard APIs share the same Puma web server process pool. Under IoT burst conditions the ingestion workload competes with dashboard requests for web worker threads, degrading dashboard latency precisely when building owners most need real-time visibility.

The application's background job layer uses Sidekiq with Redis as the queue backend. Notification fan-out, telemetry aggregation, and scheduled reporting share the same Sidekiq worker pool with no queue priority separation. During the three production incidents reviewed, notification jobs backed up behind lower-priority aggregation jobs, delaying critical leak alerts by between 4 and 22 minutes.

IoT Ingestion Path

IoT devices report telemetry via MQTT to an AWS-hosted MQTT broker connected to a Rails endpoint via a custom subscriber process. That process writes each telemetry message synchronously to a PostgreSQL table. At approximately 900 messages per second during peak hours, this produces observable write amplification: each telemetry row triggers index updates across three indices, and the primary telemetry table has grown to approximately 380 gigabytes with no partitioning.

Telemetry flow analysis, March 2025. Peak message rate measured at 897 messages per second on March 14, 2025 during a freeze-thaw weather event.

Notification Trigger Layer

The notification service fans out alerts to eight channels synchronously. A slow or unresponsive third-party channel API blocks the notification thread for the full HTTP timeout period. During the January 2025 incident, alert delivery latency exceeded 18 minutes for a subset of critical leak events.

Data Layer

A single PostgreSQL 13 instance carries the entirety of the application's data. At 380 gigabytes for the telemetry table alone, growing at approximately 55 gigabytes per month, the database

will approach one terabyte within twelve months. Performance analysis shows that time-range queries over the telemetry table account for 61 percent of all slow queries.

Code repository inspection findings. Slow query log analysis, February 2025.

Security Posture

The most significant finding concerns multi-tenant data isolation. Tenant scoping is enforced via application-level current_tenant context only -- there are no database-level row security policies. IoT device authentication uses a shared MQTT credential per site type rather than per-device X.509 certificates.

Current-State Architecture Diagram

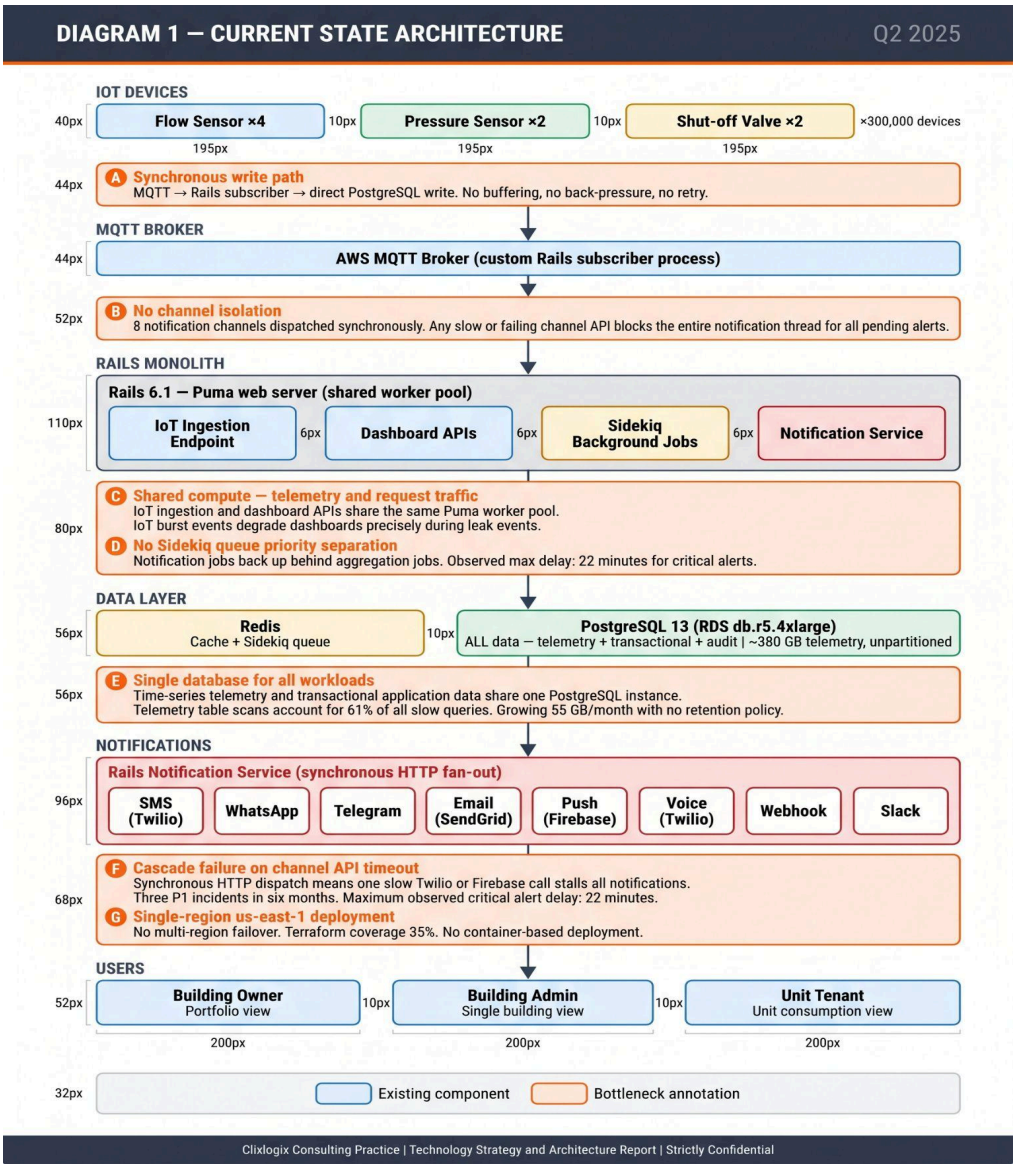

Diagram 1: Current-State Architecture — four primary bottlenecks annotated in orange

Preserve / Replace / Retire Summary

Component	Disposition	Rationale
Rails application (core business logic)	Preserve; decompose incrementally	Business logic is well-encapsulated. Domain service extraction preserves investment.
PostgreSQL (transactional data)	Preserve; partition and harden	Correct choice for transactional workloads. Requires RLS hardening.
Redis (caching and session)	Preserve; migrate to ElastiCache managed	Sound choice. Managed service reduces operational overhead.
Sidekiq (background jobs)	Replace notification path; retain for non-critical async	Notification fan-out must move to event-driven architecture.
MQTT ingestion endpoint (Rails-based)	Replace	Synchronous Rails MQTT subscriber is the primary ingestion bottleneck.
Telemetry data in PostgreSQL	Migrate to TimescaleDB	Time-series data requires a purpose-built time-series store.
Capistrano deployment	Replace with EKS/Kubernetes	Container-based deployment is a prerequisite for multi-region.
Manual Terraform state	Remediate	All infrastructure must be in version-controlled Terraform before Phase 2.
Per-site MQTT credentials	Replace with per-device X.509 certs	Shared credentials cannot be individually revoked.
CloudWatch-only observability	Augment with Datadog full stack	Distributed tracing requires a more capable observability platform.

Structured Findings

#	Finding	Area	Current Impact	Impact at 10K Buildings	Disposition
F-01	Synchronous IoT-to-PostgreSQL write path	IoT Ingestion	p99 latency >1,400ms at peak	Fails above ~2,200 msg/sec; target 8,500	Replace with Kafka-based async path
F-02	Telemetry and application traffic sharing compute	Application	Dashboard spikes during IoT burst	Resource contention blocks both workloads	Separate to dedicated compute pools

#	Finding	Area	Current Impact	Impact at 10K Buildings	Disposition
F-03	Synchronous notification fan-out in Sidekiq	Notifications	3 P1 incidents; max 22-min delay	Queue saturates at 3x current building count	Rebuild as event-driven architecture
F-04	Telemetry data in PostgreSQL without partitioning	Data Layer	61% of slow queries from telemetry scans	Performance below threshold at ~700GB	Migrate to TimescaleDB
F-05	Single-region, single-AZ deployment	Infrastructure	Acceptable at current scale	99.9%+ SLA requires multi-region active-active	Multi-region Phase 3
F-06	Application-level-only tenant isolation	Security	No confirmed breach; violates SOC 2	Must resolve before diligence	Add database row-level security
F-07	Shared MQTT credentials per device type	IoT Security	No per-device revocation	Compromised device exposes entire cohort	Migrate to per-device X.509 certs
F-08	No distributed tracing	Observability	Root-cause analysis takes hours	Cannot diagnose telemetry path end-to-end	Add Datadog distributed tracing
F-09	35% false-positive alert rate	Observability	Alert fatigue; delayed genuine incident response	Worsens with more devices and rules	Tune alerts; add anomaly-based alerting
F-10	Terraform coverage at 35%	Operations	Configuration drift; unreliable rollback	Cannot operate multi-region without full IaC	Complete Terraform coverage in Phase 1
F-11	No OTA firmware update management	Hardware Fleet	Manual coordination; no rollback	At 1M devices, manual management untenable	Implement OTA update platform in Phase 3
F-12	No time-series retention or downsampling	Data Layer	Growing at 55GB/month	2TB+ within 18 months	Implement retention policies in TimescaleDB

SECTION 4

Target State Architecture

The target-state architecture separates the platform into eleven logical layers, each with a clearly scoped responsibility, independent scaling characteristics, and defined data contracts with adjacent layers.

Architectural Principles

- **Separation of telemetry-path from request-path workloads.** IoT telemetry ingestion, real-time processing, and time-series storage operate on dedicated compute and data resources completely isolated from the user-facing application request path.
- **Async event-driven core.** Every inter-service communication that does not require a synchronous response uses events on the streaming layer.
- **Multi-region active-active for the application platform.** The application services layer and transactional data layer deploy in two AWS regions (us-east-1 and us-west-2) in an active-active configuration with global routing.
- **Zero-trust security model.** Every service-to-service call carries a verifiable identity token. Every database connection operates under a least-privilege IAM role. Every IoT device authenticates with a unique X.509 certificate.
- **Tenant isolation at every layer.** Multi-tenancy is enforced at database (row-level security), application (request-scoped tenancy enforcement), streaming (tenant-partitioned Kafka topics), and observability (tenant-scoped metrics and log streams) layers.
- **Observable by design.** Distributed tracing spans the full path from device to dashboard. Performance budgets are defined and alarmed.

Target-State Architecture Diagram

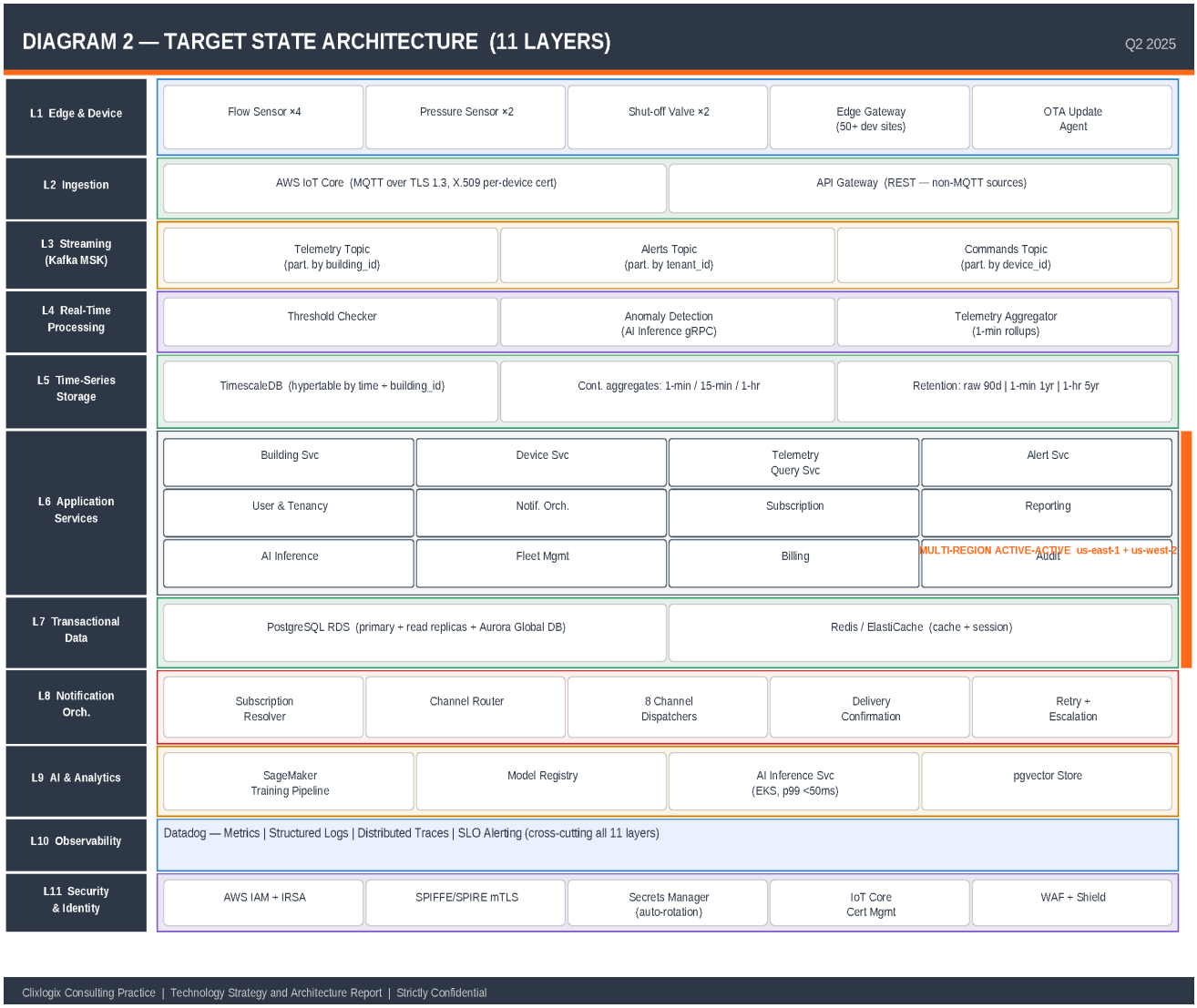

Diagram 2: Target-State Architecture — 11 layers from Edge and Device through Security and Identity

Layer-by-Layer Description

Layer 1: Edge and Device Layer

IoT devices operate at the building edge, each carrying a unique X.509 certificate. Edge gateway devices aggregate telemetry from multiple sensors for high-density sites, providing local buffering during connectivity interruptions and enabling local shut-off valve commands without cloud round-trip.

Layer 2: Ingestion Layer

AWS IoT Core replaces the custom Rails MQTT subscriber. IoT Core rules route telemetry messages to MSK Kafka via a Lambda enrichment function that validates device registration, adds ingestion timestamp, and enriches messages with tenant_id.

Layer 3: Streaming Layer

Apache Kafka on AWS MSK is the central nervous system of the telemetry processing path. Three primary topic groups: telemetry stream (partitioned by building_id), alert stream (partitioned by tenant_id), and command stream (partitioned by device_id). Replication factor of 3 provides durability.

Layer 4: Real-Time Processing Layer

Kafka Streams consumers on EKS perform threshold-based alert generation, anomaly detection inference (calling the AI Inference Service via gRPC), and telemetry aggregation for the dashboard hot path.

Layer 5: Time-Series Storage Layer

TimescaleDB on RDS PostgreSQL stores all device telemetry. The primary hypertable is partitioned by time and building_id. Continuous aggregates pre-compute one-minute, fifteen-minute, and one-hour rollups. Raw telemetry retained 90 days; one-minute aggregates for one year; one-hour aggregates for five years.

Layer 6: Application Services Layer

Twelve domain services on Amazon EKS: Building Service, Device Service, Telemetry Query Service, Alert Service, User and Tenancy Service, Notification Orchestration Service, Subscription Service, Reporting Service, AI Inference Service, Fleet Management Service, Billing Service, and Audit Service. Each service owns its own data store.

Layer 7: Transactional Data Layer

PostgreSQL on RDS with row-level security enforcing tenant isolation within each schema. Aurora Global Database provides sub-second replication lag to the secondary region and automatic promotion for failover.

Layer 8: Notification Orchestration Layer

The Notification Orchestration Service consumes alert events from Kafka and resolves the full recipient set. Eight independent channel dispatchers handle delivery with isolated retry logic and delivery confirmation.

Layer 9: AI and Analytics Layer

SageMaker training pipeline, model registry, on-EKS AI Inference Service (p99 under 50ms), and pgvector vector store for consumption pattern similarity search.

Layer 10: Observability Layer

Datadog collects metrics, logs, and distributed traces from every layer. Distributed trace context propagates from device connection through Kafka to processing and from alert through the notification path to delivery confirmation.

Layer 11: Security and Identity Layer

AWS IAM with IRSA for pod-level least-privilege. SPIFFE/SPIRE or AWS-native mTLS for service-to-service authentication. AWS Secrets Manager with automatic rotation for database credentials and API keys.

SECTION 5

IoT Data Flow Architecture

The IoT data flow describes the complete path from a sensor reading at a building device to a time-series data point queryable by a dashboard user and, where the reading triggers an alert, to a delivered notification. This path is the operational core of the Client's business.

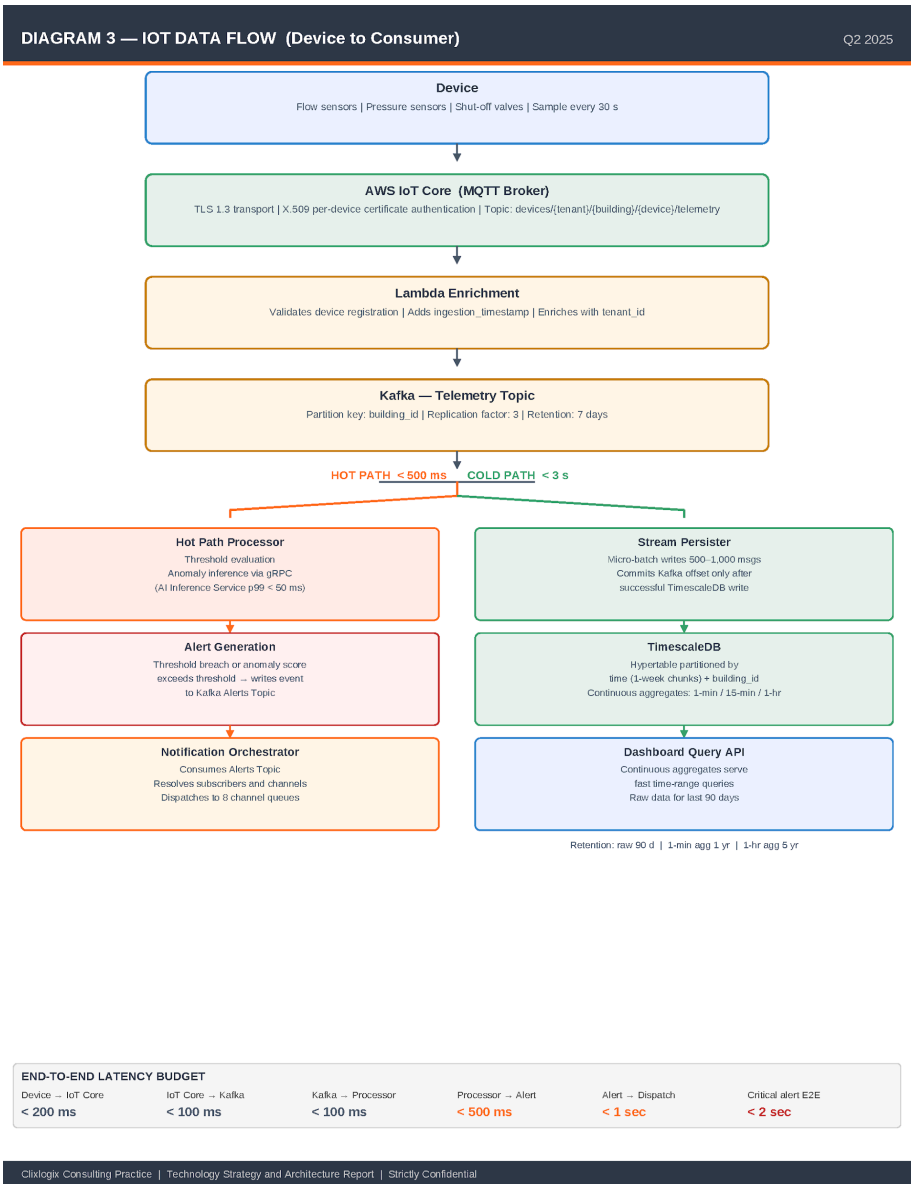

Diagram 3: IoT Data Flow — Device to Consumer, showing hot-path and cold-path branches and latency budgets

Device to Cloud Protocol and Authentication

Devices communicate with AWS IoT Core using MQTT 3.1.1 over TLS 1.3. Each device authenticates using a unique X.509 certificate provisioned during manufacturing. Certificate rotation is managed by the Fleet Management Service, which issues replacement certificates with a 30-day overlap window before expiry. The topic hierarchy follows: devices/{tenant_id}/{building_id}/{device_id}/telemetry.

MQTT Broker to Kafka: Ingestion Gateway Design

AWS IoT Core rules route each incoming MQTT message to an MSK Kafka topic via an AWS Lambda function. The Lambda function adds an ingestion_timestamp, validates that the device is registered in the Device Service, and enriches the message with the tenant_id before writing to Kafka. This design adds approximately 30 to 50 milliseconds of processing time but provides validation and enrichment that would otherwise need to happen downstream.

Assumption: Lambda cold start latency is not a concern for this ingestion path because the Lambda function remains warm under continuous production load.

Stream Partitioning Strategy

The telemetry Kafka topic uses building_id as the partition key. This ensures all telemetry from a single building is processed by the same consumer partition, enabling stateful processing (anomaly detection requires a rolling window of readings from the same building) without cross-partition coordination overhead. The alerts topic uses tenant_id as the partition key. The commands topic uses device_id, ensuring ordered delivery.

Hot Path Versus Cold Path

Hot path: A Kafka Streams consumer performs threshold evaluation, anomaly inference via gRPC to the AI Inference Service (p99 under 50ms), and one-minute aggregate update. Alert events written to the Kafka alerts topic. Target end-to-end: less than 500 milliseconds from ingestion to alert generation.

Cold path: A separate stream persister consumer writes every telemetry message to the primary TimescaleDB hypertable in micro-batches of 500 to 1,000 messages, committing Kafka offsets only after successful database write. Target cold-path persistence lag: less than 3 seconds.

Downsampling and Retention Strategy

TimescaleDB continuous aggregate policies maintain pre-computed rollups at multiple time granularities. Retention policies drop raw telemetry chunks older than 90 days. One-minute aggregates retained for one year. One-hour aggregates retained for five years. This three-tier structure reduces the active storage footprint by approximately 82 percent compared to indefinite raw retention.

SECTION 6

Notification Trigger Layer Redesign

The current notification architecture is the highest-priority reliability risk in the Client's platform. Three production incidents in six months, with maximum alert delivery delays of 22 minutes for critical leak events, represent a direct threat to the Company's contractual SLAs.

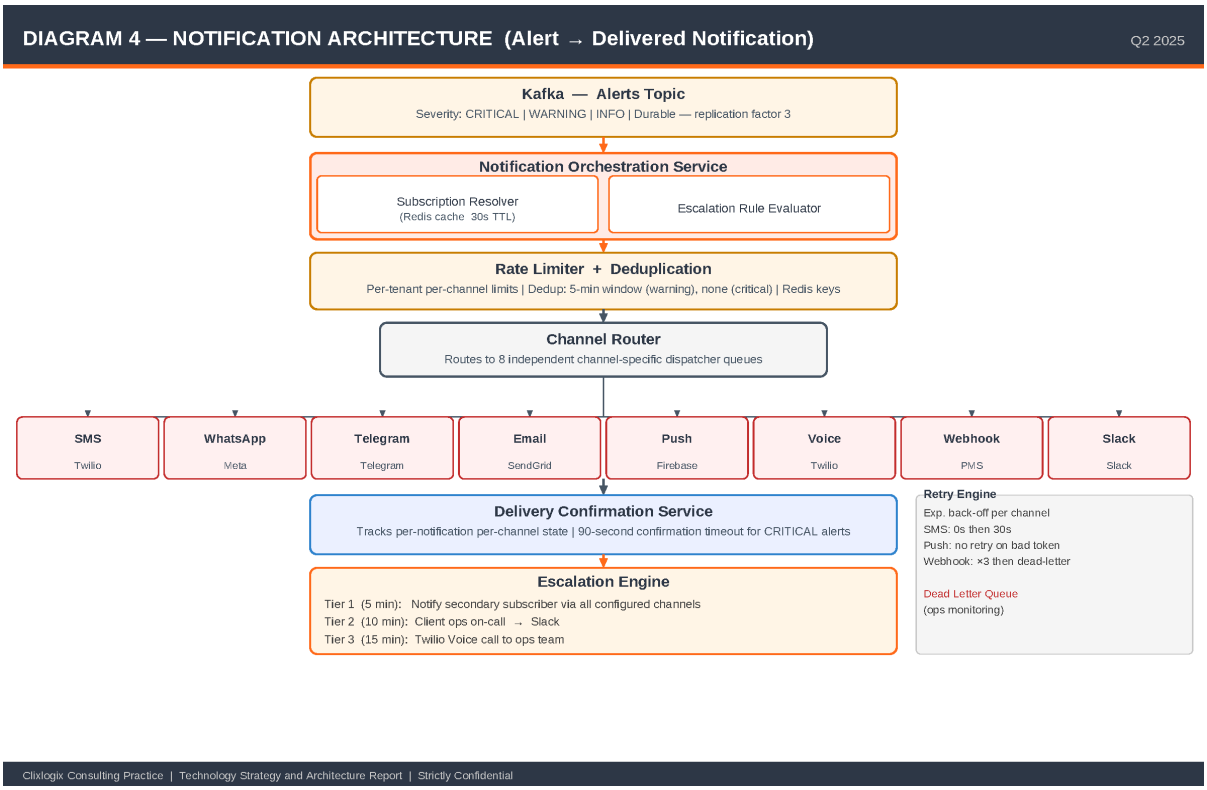

Diagram 4: Notification Architecture — Alert to delivered notification, with escalation path and channel isolation

Event Source Design

Alert events are written to the Kafka alerts topic with a structured schema: alert_id, tenant_id, building_id, device_id, alert_type, severity, triggered_at, triggering_value, threshold_value. The Kafka alerts topic provides durability: a critical leak alert is persisted to disk with replication factor 3 before the producing process acknowledges it.

Subscription Preference Resolution

The Notification Orchestration Service calls the Subscription Service to resolve the full recipient list: which users are subscribed for this building, what channels they prefer, and what escalation

rules apply. Subscription resolution is cached in Redis with a 30-second TTL. Cache invalidation occurs on any subscription preference change.

Channel-Specific Dispatcher Pattern

The channel router distributes notification payloads to eight channel-specific dispatcher queues. Each dispatcher is independent: a Twilio outage cannot delay WhatsApp or email notifications for the same alert. Each dispatcher implements: receive payload, attempt delivery, record result, retry with exponential back-off, dead-letter after three retries, emit delivery status to the Delivery Confirmation Service.

Rate Limiting and Deduplication

Rate limiting is enforced per tenant per channel. Critical alerts bypass rate limits entirely. Deduplication prevents a user from receiving the same alert via the same channel more than once within a configurable window (default five minutes for warnings, no deduplication for critical alerts). Deduplication keys are stored in Redis with TTL equal to the deduplication window.

Escalation Rules for Critical Leak Events

Tier 1 (initial trigger): all critical-severity subscribers notified via all configured channels simultaneously. Tier 2 (5-minute SLA breach): secondary subscriber tier notified; Client ops on-call notified via Slack. Tier 3 (second SLA breach): Client operations team receives a Twilio Voice call. This ensures a critical leak event at 3 AM reaches the building owner and, if necessary, the Client's own operations team within a contractually defensible time window.

SECTION 7

Technology Selection

Technology selections are made for the Client's specific situation: a North American IoT-SaaS platform at $20M ARR scaling to $60M ARR, with a team of 35 engineers, an existing AWS deployment, and a hard requirement to deliver multi-region reliability, event-driven telemetry processing, and SOC 2 Type II compliance within 18 months.

Cloud Platform

AWS is recommended. The Client's existing platform runs on AWS, the engineering team has AWS operational familiarity, and migrating to a different cloud provider during an 18-month architectural transformation would introduce a migration workload that directly competes with the modernization program for engineering capacity. AWS provides IoT Core, MSK, RDS, ElastiCache, SageMaker, and EKS -- all required managed services.

Compute and Orchestration

Criterion	Weight	EKS (Kubernetes)	ECS (Fargate)	EC2 with Auto Scaling
Multi-region deployment support	High	Excellent: standard multi-cluster patterns; Helm charts portable across regions	Good: requires ECS-specific tooling per region	Poor: custom automation required
Operational flexibility for polyglot services	High	Excellent: standard Kubernetes API; portable workloads	Good: simpler but less flexible	Poor: service-level deployment requires significant custom work
Team learning curve	Medium	Moderate: Kubernetes complexity addressable with managed node groups	Low: simpler mental model	Low: familiar EC2 model; migration downgrade
Ecosystem (Helm, KEDA, Argo CD, service mesh)	High	Excellent: full ecosystem available	Limited: AWS-native tooling only	Limited: no native service mesh
Recommendation		Selected	Not selected	Not selected

Streaming Layer

The streaming layer selection is the most consequential technology decision in this architecture. The Client's IoT telemetry workload -- high volume, ordered per device, with both hot-path real-time processing and cold-path durable storage requirements -- is precisely the workload Apache Kafka was designed for.

Criterion	Weight	Apache Kafka (MSK)	Amazon Kinesis	Amazon SQS/SNS
Throughput at target scale (8,500 msg/sec)	Critical	Excellent: MSK clusters handle millions of messages/sec; no ceiling concern	Good: Kinesis handles this but with higher per-shard cost at scale	Poor: lacks stream replay, log compaction, and consumer group features
Message replay and reprocessing	High	Excellent: Kafka log is replayable from any offset; critical for data migration	Limited: 7-day retention; no log compaction	None: SQS messages are consumed and deleted
Consumer group isolation	High	Excellent: multiple independent consumer groups at their own offsets	Moderate: multiple consumers with throughput limits per shard	None: message is consumed once
Stream partitioning by key	High	Excellent: partition key routing is native; building_id partitioning is straightforward	Moderate: partition key routing available but shard management is manual	None: no key-based routing
Vendor lock-in	Low	Low: open-source; portable to self-managed if needed	High: Kinesis API is AWS-proprietary	High: AWS-proprietary
Recommendation		Selected	Not selected: insufficient replay/reprocessing	Not selected: wrong tool for streaming workload

Time-Series Database

Criterion	Weight	TimescaleDB	InfluxDB (Cloud)	Amazon Timestream
SQL compatibility with existing PostgreSQL skills	High	Excellent: PostgreSQL extension; developers can query immediately	Poor: InfluxQL and Flux are proprietary	Moderate: SQL-like syntax with differences requiring adaptation
Continuous aggregates and retention policies	High	Excellent: native materialized views auto-refresh; built-in retention policies	Good: downsampling available; less flexible	Good: scheduled queries; retention built-in

Criterion	Weight	TimescaleDB	InfluxDB (Cloud)	Amazon Timestream
Compression at scale	High	Excellent: 90-95% storage reduction verified at multi-TB deployments	Good: vendor-managed compression	Good: Timestream manages storage tiering
Integration with existing PostgreSQL ecosystem	High	Excellent: same RDS cluster; pgvector alongside; familiar ops tooling	Poor: separate system; new monitoring and backup discipline	Moderate: AWS-native but not PostgreSQL-compatible
Vendor lock-in	Medium	Low: open-source; data in PostgreSQL format	High: proprietary; complex migration path	High: AWS-proprietary
Recommendation		Selected	Not selected: query language barrier	Not selected: vendor lock-in

Observability Stack

Criterion	Weight	Datadog (full stack)	AWS-native (CloudWatch + X-Ray)	Self-hosted (Prometheus + Grafana + Jaeger)
Distributed tracing across heterogeneous stack	Critical	Excellent: single agent covers Rails, Python, Go, Kafka consumers, Lambda	Moderate: X-Ray covers Lambda well; weaker on non-AWS runtimes	Good: OpenTelemetry standard; high ops burden
IoT-to-notification end-to-end trace	Critical	Excellent: trace context threaded from MQTT message ID through Kafka to notification delivery	Poor: no native Kafka trace context propagation	Good: possible with custom instrumentation; high implementation effort
Cost at scale (10K buildings)	Medium	Higher: estimated $240K-$290K/year at target scale	Lower: estimated 40-50% lower than Datadog	Lowest: compute cost only; ops burden is material
Recommendation		Selected	Supplementary: CloudWatch retained for AWS service metrics	Not selected: ops burden incompatible with team size

AI Infrastructure

The anomaly detection model training pipeline runs on AWS SageMaker. The production inference service runs as a containerized service on EKS with the model loaded in memory, not via SageMaker Inference Endpoints, because the latency requirement (sub-50 millisecond p99) requires avoiding the additional network hop. pgvector (the PostgreSQL vector extension) is recommended for consumption pattern similarity search over a dedicated vector database, enabling similarity queries that join against building metadata without cross-system joins.

SECTION 8

Migration and Integration Approach

The architectural transformation described in this report is executed alongside a live production system serving 3,000 buildings and approximately $20M ARR. No architectural component is replaced with a hard cutover. Every migration uses a shadow-mode, dual-write, or progressive traffic shift pattern that allows the new component to be validated under production conditions before the legacy path is decommissioned.

Core Migration Principles

- **Build beside, not instead of.** New architectural components are built and validated alongside the existing platform. The legacy path carries production traffic while the new path proves itself.
- **Shadow mode before primary mode.** Every new path runs in shadow mode -- receiving a copy of production traffic and processing it without affecting production outcomes -- for a defined validation period before being designated the primary path.
- **Dual-write with reconciliation.** During the telemetry data migration, the ingestion path writes to both PostgreSQL and TimescaleDB. A reconciliation process validates consistency before the PostgreSQL telemetry path is retired.
- **Feature flag control.** Production traffic shifts are controlled by feature flags, not deployment events. Each building's telemetry path can be shifted independently, allowing a gradual rollout with rollback capability at the building level.

IoT Ingestion Path Migration

Phase 1 establishes the Kafka-based ingestion path alongside the existing Rails MQTT subscriber. For the first 30 days, the IoT Core rule writes each incoming MQTT message to both the Rails subscriber (existing path) and the Kafka telemetry topic (new path). The Rails subscriber continues to write to PostgreSQL as before. A shadow consumer reads from the Kafka topic and writes to TimescaleDB without affecting the production alert or dashboard path.

The reconciliation process compares the PostgreSQL telemetry records and the TimescaleDB records for matching device readings on a 15-minute lag. Discrepancies are logged and reviewed. When the reconciliation error rate drops below 0.01 percent for 7 consecutive days, the feature flag is advanced to shift dashboard telemetry queries to the TimescaleDB path for a subset of buildings (initially 5 percent, growing to 50 percent, then 100 percent over a 4-week ramp).

Assumption: A 0.01 percent reconciliation error rate represents approximately 3 messages per hour at current peak load. This threshold is set to be achievable within the Phase 1 validation window while providing sufficient confidence before the PostgreSQL path is decommissioned.

Telemetry Data Migration

The existing 380-gigabyte PostgreSQL telemetry table is migrated to TimescaleDB in a background process that does not affect production. The migration reads from PostgreSQL in time-ordered batches, writes to TimescaleDB via the standard ingestion path, and maintains a watermark that allows restartable progress. The migration is estimated to complete in 72 to 96 hours at the sustained batch write rate achievable without affecting production PostgreSQL performance.

Historical telemetry older than 90 days is migrated to TimescaleDB and, after a 30-day validation window, deleted from PostgreSQL. Data between 0 and 90 days is available in both systems during the dual-write period; after decommissioning, it is available only in TimescaleDB.

Notification Layer Migration

The notification layer migration is a Phase 2 workstream and does not begin until the streaming layer is stable and the team has operational experience with Kafka consumer patterns. The migration follows a channel-by-channel approach: the SMS channel is migrated first (lowest risk, most straightforward dispatcher), followed by email, push, and finally WhatsApp and voice. Each channel migration includes a shadow-mode period where both the legacy Sidekiq path and the new dispatcher produce notifications, but only one (initially the legacy path, then the new path) actually delivers them.

The feature flag for each channel is controlled per account, allowing the migration to be limited to internal test accounts and then design-partner accounts before full rollout. The existing Sidekiq notification path is decommissioned channel-by-channel only after 30 days of production operation of the new dispatcher with zero P1 incidents.

API Design Principles

All new services expose versioned REST APIs (v1 prefix, versioned by URL path segment) with OpenAPI 3.0 specifications generated from code. APIs are designed around resources (buildings, devices, alerts, users, subscriptions) with predictable RESTful semantics. Internal service-to-service APIs use gRPC for the performance-critical paths (AI Inference Service, Subscription Service) and REST for the remainder.

During the migration period, the existing Rails API endpoints continue to function and are backed initially by the Rails application, then progressively by the new domain services as those services are validated in production. This approach preserves backward compatibility for existing mobile and web clients throughout the migration.

Migration Phasing Diagram

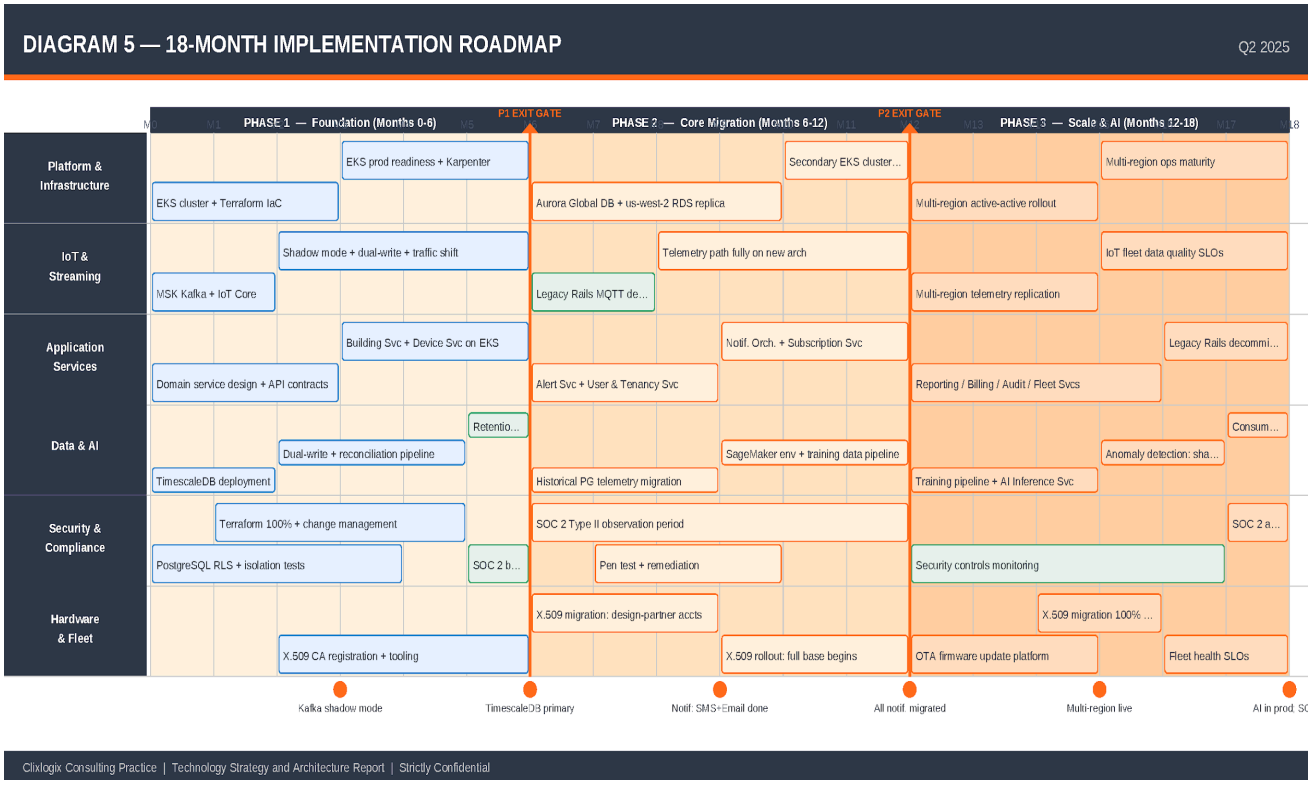

Diagram 5: Migration and implementation phasing — three phases with workstream bars, phase gates, and key milestones

SECTION 9

Scalability and Performance

Capacity projections are extrapolated from the performance baseline measured in March 2025 at 3,000 buildings and 300,000 devices, applied to the target scale of 10,000 buildings and approximately 1,000,000 devices. Projections assume a linear relationship between building count and message volume, which the telemetry flow analysis supports within a margin of approximately 15 percent. Non-linear effects (weather events causing correlated alert bursts across many buildings) are addressed in the burst capacity assumptions.

Load Projections at 10,000 Building Scale

Metric	Current (3K buildings)	Projected (6K buildings)	Projected (10K buildings)	Architecture Capacity Target
Peak IoT messages per second	~900 msg/sec (measured peak)	~1,800 msg/sec	~8,500 msg/sec (incl. burst factor 1.7x)	MSK: designed for >50,000 msg/sec on 3-broker cluster
Alert events per minute (normal)	~120 alerts/min	~240 alerts/min	~400 alerts/min	Notification layer: 5,000 alerts/min before rate limiting
Alert events per minute (burst)	~1,400 alerts/min (observed peak)	~2,800 alerts/min	~4,700 alerts/min	Notification layer burst capacity: 15,000 alerts/min
Dashboard concurrent users (peak)	~1,200 users	~2,400 users	~4,000 users	EKS application layer: 15,000 concurrent users with horizontal scaling
API requests per second	~380 req/sec	~760 req/sec	~1,270 req/sec	API Gateway + EKS: 10,000 req/sec with HPA scaling
Time-series data volume per day	~180 GB/day (raw)	~360 GB/day	~1,050 GB/day (at 10K buildings)	TimescaleDB: 10+ TB/day documented; compression reduces effective write ~90%
Cumulative time-series (active, 90-day raw)	~380 GB (current)	~760 GB	~950 GB (post-retention-policy active footprint)	TimescaleDB on r5.4xlarge RDS: 4 TB practical ceiling before read replica needed

Metric	Current (3K buildings)	Projected (6K buildings)	Projected (10K buildings)	Architecture Capacity Target
Notification deliveries per day	~45,000/day	~90,000/day	~150,000/day	Channel dispatcher pool scales horizontally with queue depth

Projected values extrapolated from Telemetry flow analysis, March 2025. Burst factor of 1.7x applied to IoT message rate reflects observed ratio of peak-to-average during weather-event days in the review period.

Performance Budgets

SLO / Performance Target	Target Value	Current Measured Value	Gap	Architecture Mechanism
Telemetry ingestion latency (p50)	<100ms	180ms (p50 at peak)	80ms gap to close	Kafka async ingestion eliminates synchronous PostgreSQL write latency
Telemetry ingestion latency (p99)	<500ms	>1,400ms at peak	900ms+ gap to close	Kafka ingestion is non-blocking; p99 decoupled from database write latency
Alert end-to-end latency (critical, p50)	<2 seconds	~6 seconds average (non-incident)	~4 second gap	Hot-path processor + event-driven notification dispatcher
Alert end-to-end latency (critical, p99, non-incident)	<5 seconds	Not reliably measured (no distributed tracing)	Unmeasurable currently	Datadog distributed tracing enables SLO measurement and enforcement
Alert delivery during incident (worst case)	<15 minutes	Up to 22 minutes (observed)	Eliminate this failure mode	Channel isolation: no channel failure can delay another channel
Dashboard API response (p95)	<800ms	~1,100ms for portfolio views	~300ms gap	TimescaleDB continuous aggregates; read replicas for portfolio queries
Platform availability (data plane)	99.9% monthly	Not formally measured	SLO undefined currently	Multi-region active-active; Aurora Global Database failover
Platform availability (control plane)	99.5% monthly	Not formally measured	SLO undefined currently	EKS with multi-AZ node groups; rolling deployments
AI inference latency (anomaly detection, p99)	<50ms per building window	Not applicable (not in production)	New capability	On-EKS inference service; model loaded in memory; no network round-trip to SageMaker

Capacity Planning by Component

Component	Current Capacity	Capacity at 10K Buildings	Scaling Mechanism	Lead Time to Scale
MSK Kafka (3-broker cluster)	Not yet deployed	3x r5.2xlarge brokers; 3 TB EBS each	Add brokers or increase EBS; partition rebalancing required	2-4 hours (broker add); 1-2 days (partition rebalance)
TimescaleDB (RDS)	380 GB active (PostgreSQL)	950 GB active raw + 120 GB aggregates	Vertical scaling to r5.8xlarge; read replica for analytical queries	Minutes for read replica add; 30-min maintenance window for vertical scale
EKS application services	Not yet deployed	12 services; HPA on CPU/memory; 3-8 pods per service baseline	Karpenter auto-provisioning of nodes; HPA scales pods	Seconds to minutes (pod scale); 3-5 minutes (new node)
RDS PostgreSQL (transactional)	db.r5.4xlarge, 1 AZ	db.r5.8xlarge, 2 AZ + read replicas + Aurora Global	Aurora Global Database auto-storage scaling; add read replicas	Hours for instance class change; minutes for replica add
Notification dispatcher pool	Sidekiq workers (shared)	Per-channel EKS deployments; HPA on queue depth	KEDA (Kubernetes Event-Driven Autoscaler) scales on Kafka consumer lag	Seconds to minutes
ElastiCache Redis	Single node, manual failover	Cluster mode enabled; 3-shard primary + read replicas	ElastiCache auto-scaling; shard add for capacity	Minutes to hours

Pressure-Testing Assumptions

The capacity projections assume that building growth is distributed across the geographic portfolio rather than concentrated in a single AWS region or Availability Zone. A concentration scenario (for example, a major contract win adding 2,000 buildings in a single month) would stress the onboarding path and the initial telemetry ingestion capacity, but not the steady-state operating capacity.

Burst assumptions are based on the observed correlation between weather events and alert volume. A significant freeze event affecting buildings across multiple markets simultaneously represents the P99 burst scenario. The notification layer capacity targets (15,000 alerts per

minute burst capacity) are sized to handle a concurrent-freeze scenario affecting 30 percent of the building portfolio simultaneously, which exceeds any observed event in the review period.

SECTION 10

Security and Multi-Tenant Isolation

Security for the Client's platform operates across three distinct risk surfaces: the IoT device and connectivity surface (where physical devices in buildings communicate with the cloud), the application and data surface (where tenant data must remain isolated across multi-tenant infrastructure), and the compliance surface (where the Company must demonstrate a defined security posture to Series C investors and, ultimately, to SOC 2 Type II auditors).

Threat Model Summary

Threat Category	Specific Threat	Current Exposure	Target-State Mitigation
Device-side	Compromised device credential used to inject malicious telemetry	Shared credential per device type; no per-device revocation	Per-device X.509 certs; IoT Core certificate revocation; ingestion validation in Lambda enrichment layer
Device-side	Physical device tampering to extract credentials	Credential stored in device firmware; no HSM	HSM for new device hardware revisions; firmware signing and integrity verification
Ingestion path	Replay attack: replaying captured telemetry messages	No message deduplication or timestamp validation at ingestion	Ingestion Lambda validates message_id uniqueness (Bloom filter, 24-hour window) and timestamp recency
Ingestion path	Message injection via stolen device certificate	Certificate revocation not automated	AWS IoT Core certificate revocation via CRL; Fleet Management Service triggers revocation within 60 seconds of tamper signal
Application	Tenant A accessing Tenant B's data via API manipulation	Application-level tenancy enforcement only; no DB row security	PostgreSQL row-level security on all tenant-scoped tables; enforced at DB regardless of application context
Application	Privilege escalation by building admin to portfolio owner	Role-based access control implemented but not	RBAC redesign as part of User and Tenancy Service; privilege matrix reviewed in SOC 2 preparation

Threat Category	Specific Threat	Current Exposure	Target-State Mitigation
		formally reviewed	
Application	API key or JWT token exfiltration and replay	JWT signing key not rotated; API keys stored in application config	Secrets Manager with automatic rotation; JWT short-lived (15-minute expiry) with refresh token rotation
Data leakage	Observability data exposing cross-tenant information	Shared log streams; metrics not tenant-scoped	Tenant ID in all log events; Datadog restrictions by tenant for operator-accessible dashboards
Compliance	SOC 2 audit finding on multi-tenant isolation	Application-only isolation does not meet SOC 2 CC6.3 standard	Database-level RLS plus documented isolation architecture satisfies CC6.3 requirement

Multi-Tenant Isolation Design

Database Layer

Row-level security (RLS) policies on all tenant-scoped PostgreSQL tables enforce that every query -- regardless of application context -- returns only rows belonging to the authenticated tenant. Each RLS policy takes the form: USING (tenant_id = current_setting('app.current_tenant_id')). The application sets the current_tenant_id session variable at the start of every database connection from the request context, and the database enforces the restriction at the storage engine level.

For large enterprise tenants (defined as more than 500 buildings or accounts with specific data residency requirements), a schema-per-tenant approach is available as an upgrade from RLS, providing complete physical isolation within the same PostgreSQL cluster. This upgrade path is documented but not required for the current customer base; no tenant currently meets the threshold that would justify the operational overhead of per-tenant schema management.

Application Layer

Every inbound API request passes through an authentication middleware that validates the JWT, extracts the tenant_id and role claims, and sets the request context. Downstream services receive the tenant context via a propagated header (X-Tenant-ID) and validate it against the JWT claim before processing. Services that call other services over internal APIs forward the tenant context in the inter-service call, which the receiving service validates independently.

Streaming Layer

Kafka topic partitioning by tenant_id (for the alerts topic) ensures that tenant-scoped consumers receive only their own tenant's events. The Notification Orchestration Service validates the tenant_id on every consumed event against the subscription resolver's output before dispatching. A consumer group per tenant is not practical at the current scale but is available as an isolation upgrade for enterprise accounts requiring dedicated processing.

Observability Layer

Every log event and metric emitted by every service carries the tenant_id as a structured field. Datadog restrictions are configured to scope operator-accessible dashboards by tenant, so that a customer success team member accessing the support dashboard for Tenant A cannot view Tenant B's telemetry or alert data. Clixlogix recommends that the Client explicitly document this observability isolation as a control in their SOC 2 Type II control framework.

SOC 2 Type II Readiness

The Client's target is SOC 2 Type II readiness by the end of Phase 2 (month 12). SOC 2 Type II requires a minimum 6-month observation period, which means the audit-ready controls must be in place by month 6 at the latest. The following controls are currently deficient and must be addressed in Phase 1:

- CC6.3 (Logical access to systems and data is restricted): Requires database-level multi-tenant isolation, not application-level only. Remediation: PostgreSQL RLS implementation in Phase 1.
- CC7.2 (Change management): Requires all infrastructure changes via reviewed and approved IaC. Remediation: Complete Terraform coverage in Phase 1.
- CC7.3 (Monitoring of system components for anomalies): Requires structured logging, distributed tracing, and anomaly-based operational alerting. Remediation: Datadog full-stack deployment in Phase 1.
- CC9.1 (Vendor risk management): Requires documented risk assessments for Twilio, SendGrid, Firebase, and other critical third-party services. Remediation: Vendor risk register created during Phase 1 security baseline work.

Series C Diligence Requirements

Series C investors in the property technology sector apply a consistent set of technical diligence criteria. Based on patterns in comparable raises, the following items are likely to be specifically assessed:

- Multi-tenant data isolation architecture documentation and evidence that database-level isolation is in place
- Incident history and post-mortem quality, specifically the three notification incidents reviewed in this engagement
- SOC 2 Type II audit status or a credible roadmap to audit completion
- Disaster recovery documentation, specifically the RTO (Recovery Time Objective) and RPO (Recovery Point Objective) for the production platform
- Penetration test results within the prior 12 months

The architecture described in this report directly addresses the first three items. Items four and five (disaster recovery documentation and penetration testing) are not architectural questions

but operational and compliance tasks that should be scoped and scheduled as part of the Phase 1 security baseline workstream.

SECTION 11

Cost Projections

Cost projections are presented across two dimensions: build cost (the engineering investment required to execute the 18-month modernization program) and run cost (the ongoing cloud, tooling, and third-party infrastructure cost at three scale points). Unit economics are presented at the end of this section to frame the cost envelope relative to the revenue the infrastructure enables.

Build Cost: 18-Month Modernization Program

Phase / Workstream	Duration	Core Team Requirement	Estimated Cost Range
Phase 1: Foundation (months 0-6)	6 months	2 platform engineers, 2 backend engineers, 1 SRE, 0.5 security engineer, 1 PM	$620K - $780K
Phase 2: Core Migration (months 6-12)	6 months	3 platform engineers, 4 backend engineers, 1 SRE, 0.5 security engineer, 1 PM	$880K - $1.1M
Phase 3: Scale and AI (months 12-18)	6 months	2 platform engineers, 3 backend engineers, 1 ML engineer, 1 SRE, 1 PM	$780K - $980K
Program management and architecture oversight	18 months	0.5 principal architect (Clixlogix advisory), 1 program manager	$320K - $420K
Security and compliance (SOC 2 prep)	18 months	External security advisory and audit preparation support	$180K - $250K
Contingency (15%)			$417K - $531K
Total build investment	18 months		$3.2M - $4.06M

Engineering costs calculated at loaded rates for the Client's hiring markets (US and Canada), including benefits, employer taxes, and management overhead. Contract engineering rates applied where internal hiring velocity is insufficient to meet phase staffing. Rates assumed at $185,000-$245,000 loaded annual cost per engineer.

Run Cost: Cloud and Infrastructure (Year 1-3)

Component	Year 1 (3K buildings)	Year 2 (6K buildings)	Year 3 (10K buildings)	Primary Cost Drivers
EKS compute (EC2, Spot mix)	$95K - $115K	$160K - $195K	$240K - $290K	Pod count scales with building count; Spot instances reduce cost 60-70% vs On-Demand for non-critical services
RDS PostgreSQL (primary + replicas)	$52K - $65K	$85K - $105K	$130K - $160K	Instance class increases at 6K buildings; read replica count increases at 10K
TimescaleDB on RDS	$35K - $45K	$60K - $75K	$95K - $120K	Storage cost dominates after year 1; compression reduces effective storage 90%
MSK Kafka (3-broker)	$38K - $48K	$55K - $70K	$80K - $100K	Broker count increases at 10K buildings for throughput headroom
AWS IoT Core	$28K - $35K	$52K - $65K	$88K - $110K	Per-message and per-connection pricing; scales linearly with device count
ElastiCache Redis	$22K - $28K	$32K - $40K	$45K - $58K	Cluster mode enabled at 6K buildings; shard count increases
Data transfer and bandwidth	$35K - $45K	$60K - $75K	$105K - $135K	Cross-AZ replication and IoT data ingress; scales with device count
SageMaker (training pipeline)	$18K - $25K	$25K - $35K	$35K - $48K	Weekly training runs; GPU instance cost depends on model complexity
Other AWS services (API GW, Lambda, S3, etc.)	$32K - $40K	$50K - $65K	$80K - $100K	Lambda enrichment function; S3 for firmware, backups, reports
Datadog (metrics + logs + traces)	$120K - $150K	$180K - $220K	$240K - $290K	Host-based pricing + log ingestion volume; largest non-AWS line item
Security tooling (pen test, secret mgmt, etc.)	$60K - $80K	$80K - $100K	$100K - $130K	Annual pen test; Secrets Manager; WAF
Third-party services (Twilio, SendGrid, Firebase)	$110K - $140K	$160K - $200K	$210K - $260K	Scales with alert volume and notification deliveries
Total annual run cost	$645K - $816K	$999K - $1.245M	$1.448M - $1.801M	

Unit Economics

Scale Point	Buildings	Annual ARR	Annual Run Cost	Run Cost % of ARR	Run Cost per Building per Month
Current (pre-modernization)	3,000	$20M	~$350K (estimated legacy infra)	~1.75%	~$9.72
Year 1 target-state	3,000	$20M	$645K - $816K	3.2% - 4.1%	$17.92 - $22.67
Year 2 target-state	6,000	$38M (est.)	$999K - $1.245M	2.6% - 3.3%	$13.87 - $17.29
Year 3 target-state	10,000	$60M	$1.448M - $1.801M	2.4% - 3.0%	$12.07 - $15.01

The unit economics show a temporary cost increase in Year 1 as the modernized infrastructure is stood up alongside the legacy system. By Year 3, the cost per building per month at target scale ($12.07 to $15.01) is competitive with comparable IoT-SaaS platforms and represents approximately 2.4 to 3.0 percent of ARR, well within the infrastructure cost benchmarks for a capital-efficient SaaS business in this segment.

Year 2 and Year 3 ARR estimates are derived from the Client's growth targets and are provided for unit economics framing only. They do not represent revenue projections.

SECTION 12

Implementation Sequence

The 18-month program is divided into three phases, each with defined scope, explicit dependencies, success criteria, and a risk gate between phases. Phases are structured so that the highest-risk work (streaming layer introduction, dual-write migration) occurs in Phase 1, where it can be validated thoroughly before Phase 2 depends on it. Phase 3 represents additive capability (multi-region activation, AI layer) that does not block the core reliability improvements delivered in Phases 1 and 2.

Phase 1: Foundation (Months 0-6)

Phase 1 establishes the architectural foundation that every subsequent phase depends on. No Phase 2 work begins until the Phase 1 exit criteria are met.

Dimension	Detail
Scope	EKS cluster and Kubernetes operational foundation; MSK Kafka deployment and shadow-mode ingestion; TimescaleDB deployment and dual-write validation; Datadog full-stack observability (metrics, logs, distributed traces); PostgreSQL row-level security implementation; Terraform infrastructure-as-code coverage to 100%; per-device X.509 certificate migration initiation; SOC 2 baseline controls implementation
Team	2 platform engineers (Kubernetes, Terraform), 2 backend engineers (ingestion path, migration tooling), 1 SRE (observability, reliability), 0.5 security engineer (RLS, SOC 2 controls), 1 PM
Key dependencies	AWS MSK and EKS service limits requested before month 1 begins; TimescaleDB RDS extension availability confirmed; Datadog contract executed; per-device X.509 cert CA registered with IoT Core
Risk gates	Kafka throughput validated at 3x current peak load before dual-write enabled; RLS implementation validated to produce zero cross-tenant query results in automated isolation test suite before production deployment
Success criteria	Kafka carrying 100% of new telemetry ingestion; TimescaleDB serving 100% of dashboard telemetry queries; Datadog distributed traces live from IoT Core to dashboard API; RLS in production on all tenant-scoped tables; Terraform coverage 100%
Exit criteria before Phase 2	All success criteria met; zero P1 incidents in 30-day stability window; reconciliation error rate below 0.01% for 7 consecutive days

Phase 2: Core Migration (Months 6-12)

Phase 2 migrates the highest-impact reliability concerns: the notification layer and the initial application service extractions. Multi-region infrastructure is prepared but not activated.

Dimension	Detail
Scope	Notification layer full redesign and channel-by-channel migration; Building Service and Device Service extraction to EKS; Alert Service and User and Tenancy Service extraction; Aurora Global Database replication configuration for us-west-2; secondary EKS cluster provisioning in us-west-2; SOC 2 Type II observation period begins; Subscription Service extraction and per-account escalation rule configuration
Team	3 platform engineers (multi-region infra, EKS), 4 backend engineers (service extraction, notification redesign), 1 SRE (release engineering, SLO definition), 0.5 security engineer (SOC 2 controls completion), 1 PM
Key dependencies	Phase 1 exit criteria fully met; Kafka stable for 30 days in production before notification layer migration begins; per-channel dispatcher design reviewed and approved before implementation begins
Risk gates	Shadow-mode notification validation: new dispatcher must achieve parity with legacy path (equal or better delivery rate) for 30 days on each channel before channel is cut over; zero P1 notification incidents in 30-day post-cutover window per channel
Success criteria	All 8 notification channels migrated to new dispatcher architecture; 4 application services running on EKS with zero legacy Rails serving those domains; Aurora Global Database replication to us-west-2 with <1-second lag verified; SOC 2 observation period begun
Exit criteria before Phase 3	Notification layer fully migrated; zero Sidekiq-based notification jobs in production; multi-region infrastructure validated in load test; SOC 2 observation period at or past month 6 of required window

Phase 3: Scale and AI (Months 12-18)

Phase 3 activates the multi-region capability, brings the AI anomaly detection layer into production, and completes the hardware fleet management modernization.

Dimension	Detail
Scope	Multi-region active-active traffic activation (progressive: 10%, 30%, 50%, 100% to us-west-2); SageMaker training pipeline for anomaly detection model; AI Inference Service deployment on EKS; anomaly detection in production for all buildings; consumption optimization recommendation beta; OTA firmware update platform deployment; per-device X.509 cert migration completion; remaining service extractions (Reporting Service, Billing Service, Audit Service, Fleet Management Service); SOC 2 Type II audit submission
Team	2 platform engineers (multi-region operations), 3 backend engineers (remaining services, AI integration), 1 ML engineer (SageMaker pipeline, model evaluation), 1 SRE (multi-region SLO, incident response), 1 PM

Dimension	Detail
Key dependencies	Phase 2 exit criteria fully met; historical telemetry data sufficient for model training (minimum 12 months of structured building data); SageMaker GPU instance quota approved; external SOC 2 auditor engaged
Risk gates	Multi-region traffic activation gated on Aurora Global Database failover test passing: failover must complete in under 30 seconds with zero confirmed data loss; AI anomaly detection model must achieve less than 5% false-positive rate in 30-day production shadow mode before becoming primary alert source
Success criteria	Multi-region active-active serving 100% of production traffic across two regions; AI anomaly detection generating alerts in production with false-positive rate below target; OTA firmware update platform covering 100% of device fleet; SOC 2 Type II audit submitted
Program completion criteria	All Phase 3 success criteria met; legacy Rails PostgreSQL telemetry table decommissioned; per-device X.509 certs covering 100% of active devices; all 12 application services on EKS; SOC 2 audit result received

Implementation Roadmap

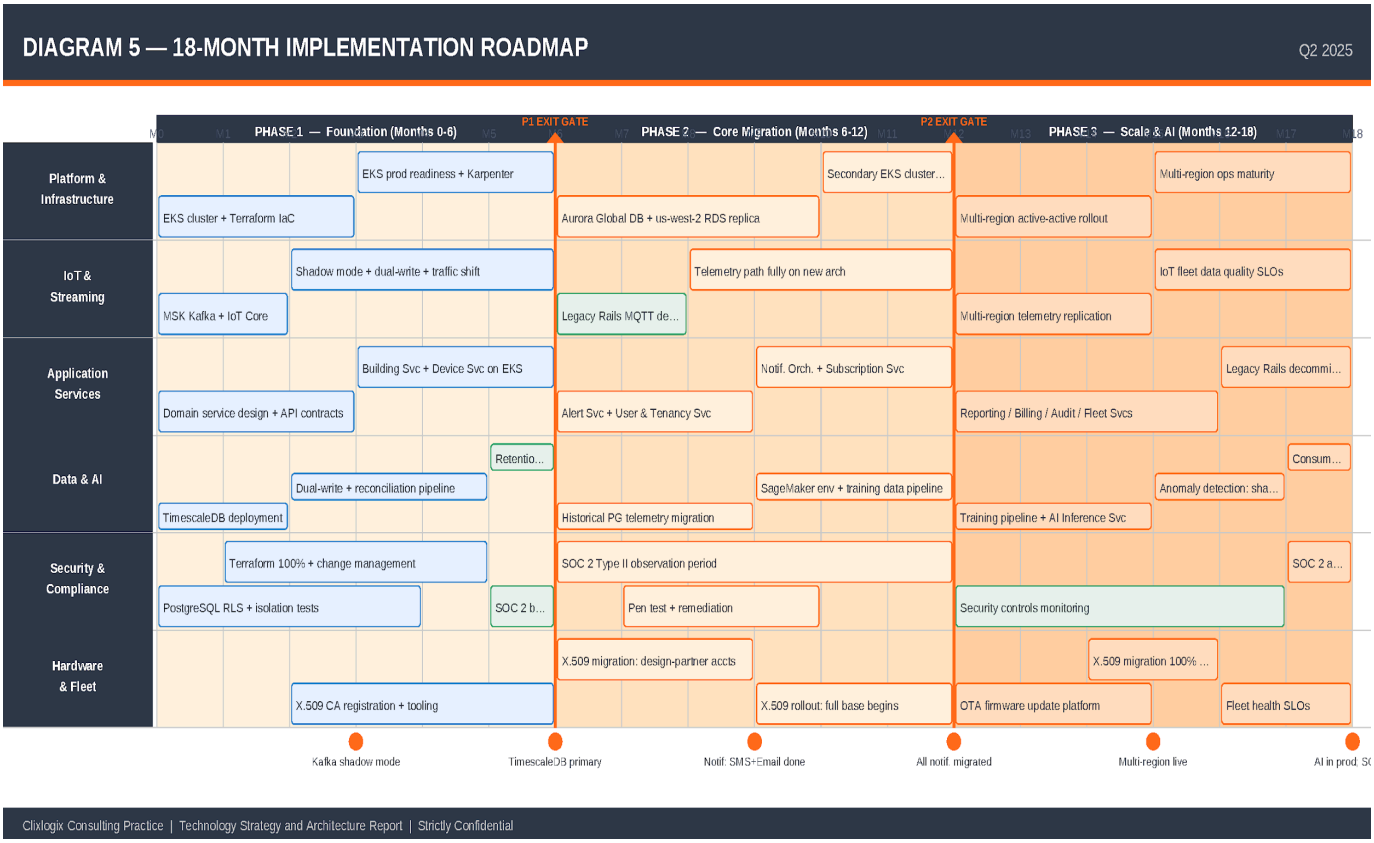

Diagram 5: 18-Month Phased Roadmap — six swim lanes across three phases with phase gates and key milestones

SECTION 13

Risk Register

Risks are scored on a 1-5 scale for both likelihood and impact. Severity is the product of likelihood and impact, yielding a range of 1 to 25. Risks with severity 15 or above are classified as high and require active management by a named owner with a defined mitigation plan.

#	Risk	Category	Likelihood (1-5)	Impact (1-5)	Severity	Owner	Mitigation
R-01	Dual-write inconsistency during PostgreSQL-to-TimescaleDB migration produces data gaps in telemetry history	Technical	3	4	12 (Medium)	Head of Platform	Reconciliation pipeline running continuously during dual-write window; 0.01% error rate threshold before legacy path decommission; 30-day validation before cutover
R-02	Kafka consumer lag accumulates during a high-burst event before the team has operational experience managing it	Technical	3	4	12 (Medium)	SRE Lead	KEDA autoscaling on consumer lag metric; runbook for emergency consumer group scaling; Datadog alert on sustained lag above 10,000 messages
R-03	EKS operational complexity causes deployment incidents during Phase 1 stabilisation	Technical	3	3	9 (Medium)	Platform Lead	Managed node groups reduce operator burden; Argo CD GitOps prevents manual deployment errors; initial services are non-critical background workloads
R-04	TimescaleDB performance degrades under high-cardinality write workload	Technical	2	4	8 (Medium)	Head of Platform	TimescaleDB chunk sizing calibrated to write pattern before go-live; compression applied from day 1; continuous aggregate refresh schedules tested at load

#	Risk	Category	Likelihood (1-5)	Impact (1-5)	Severity	Owner	Mitigation
	before tuning is applied						
R-05	Notification dispatcher for WhatsApp Business API fails due to API policy changes by Meta	Technical	2	3	6 (Low)	Head of Engineering	Dead-letter handling routes undelivered notifications to fallback channel; no single channel is critical path for any alert; API version pinning and change notification monitoring
R-06	Engineering team of 35 cannot sustain parallel feature development and architectural transformation	Program	4	4	16 (High)	CTO	Phase sequencing isolates transformation workstreams from product feature delivery; contract engineering for platform-specific roles; explicit feature freeze on the components under migration for the duration of each migration window
R-07	Key staff attrition during the 18-month program disrupts continuity in critical workstreams	Program	3	4	12 (Medium)	VP Engineering	Knowledge documentation requirements embedded in Definition of Done; pair programming on all critical path workstreams; retention risk review at each phase gate
R-08	Series C timeline accelerates, compressing Phase 1 security and SOC 2 baseline work	Program	3	4	12 (Medium)	CTO	SOC 2 baseline controls are prioritized as Phase 1 day-1 workstream, not deferred to end of phase; security work is parallel to infrastructure work, not sequential
R-09	AI anomaly detection model fails to reach acceptable false-positive	Technical	2	3	6 (Low)	ML Lead	30-day shadow mode with false-positive rate measurement before production activation; model operates in parallel with

#	Risk	Category	Likelihood (1-5)	Impact (1-5)	Severity	Owner	Mitigation
	rate threshold in production						threshold-based alerting; fallback to threshold-only if model does not meet 5% threshold
R-10	Multi-region Aurora Global Database replication lag exceeds SLO during primary region stress events	Technical	2	4	8 (Medium)	SRE Lead	Aurora Global Database replication lag monitored continuously; traffic routing policy shifts writes to primary only if lag exceeds 1 second; load test replication path before active-active activation
R-11	AWS service limit increase requests (MSK, EKS, IoT Core) are delayed and block Phase 1 start	Operational	2	3	6 (Low)	Head of Platform	Service limit requests submitted in programme week 1, before implementation begins; limit increase requests documented with business justification to accelerate AWS approval
R-12	Per-device X.509 certificate migration disrupts connectivity for a subset of devices during rollout	Technical	2	4	8 (Medium)	Head of Hardware Ops	Credential overlap window of 30 days before old credential is deactivated; migration is staged by building, not by device type; rollback procedure restores old credential without device firmware update
R-13	Datadog costs exceed budget projections if log volume grows faster than modelled	Financial	3	2	6 (Low)	Head of Platform	Log sampling applied at debug and info levels in production (only warnings and errors unsampled); Datadog log index exclusion filters configured from day 1; cost reviewed monthly against budget
R-14	SOC 2 Type II audit	Compliance	2	4	8 (Medium)	Head of	Pre-audit readiness assessment by

#	Risk	Category	Likelihood (1-5)	Impact (1-5)	Severity	Owner	Mitigation
	identifies a control deficiency that requires architectural remediation after Phase 2					Security	external security advisory in Phase 2 month 3 (prior to formal audit); all CC6 and CC7 controls implemented and tested before observation period begins
R-15	Product feature requests from the sales pipeline conflict with the architectural migration freeze on specific components	Program	4	3	12 (Medium)	CTO / VP Product	Migration windows are time-bounded and communicated to product and sales 90 days in advance; emergency exception process defined for revenue-critical features; program plan published to full leadership team at phase start

SECTION 14

Recommendation Summary

Core recommendation: Execute the 18-month, three-phase platform modernization program described in this report. Separate the IoT telemetry processing path from the application request path. Replace the synchronous PostgreSQL-based ingestion with an event-driven Kafka pipeline. Migrate telemetry data to TimescaleDB with retention and compression policies. Rebuild the notification layer as an event-driven, channel-isolated architecture. Harden multi-tenant isolation at the database layer. Activate multi-region active-active in Phase 3. Introduce AI anomaly detection as the culminating capability in Phase 3. Execute this program while the existing platform remains in production.

The Do-Nothing Alternative

Not executing this program is not a neutral option. The current architecture fails specifically at 10,000-building scale across four measurable dimensions. The IoT ingestion path is limited to approximately 2,200 messages per second at the current peak; 10,000 buildings requires 8,500 messages per second. The notification layer has already produced three P1 incidents at 3,000 buildings; at 10,000 buildings, the alert volume is 3.3 times higher and the fan-out queue saturation incidents will be more frequent and more severe. The PostgreSQL telemetry table, currently at 380 gigabytes, reaches the performance degradation threshold (approximately 600 to 700 gigabytes for unpartitioned time-range queries) within 8 months at the current growth rate. And the multi-tenant isolation gap creates a specific, non-hypothetical diligence risk: institutional Series C investors in the property technology sector consistently flag application-only tenant isolation as a control deficiency.

A company that reaches 5,000 buildings on the current architecture will spend more in engineering remediation effort -- in production, under load, with customer impact -- than the cost of executing this program in a planned, phased manner. The program cost is investment in controlled modernization. The alternative is reactive remediation at higher cost under worse conditions.

Conditions Under Which to Revisit

- **Series C does not close.** If the funding round does not close and the growth target is revised significantly downward, the multi-region and AI layer components of Phase 3 should be deferred. Phases 1 and 2 (streaming, time-series, notification reliability, security hardening) remain necessary at any growth trajectory and should proceed regardless of the funding outcome.
- **Growth trajectory is revised to below 6,000 buildings over 24 months.** The Phase 3 multi-region activation can be deferred if the business case for the additional operational

complexity changes. The architecture supports multi-region activation as an incremental step, not as a prerequisite for Phase 1 or Phase 2 benefits.

- **A regulatory change reshapes the data residency picture.** If a significant regulatory development requires Canadian customer data to remain in Canadian AWS regions (for example, a specific provincial data residency requirement affecting building IoT data), the multi-region architecture described in this report provides the foundation for a Canada-specific region deployment, but the routing and data partitioning design would require a targeted addendum to this report.

None of these conditions changes the fundamental assessment: the current architecture does not scale to 10,000 buildings, and the modernization program described here is the cost-efficient, risk-managed path to the platform the Company's growth requires.

APPENDIX

Appendix

A. Interview List (Anonymized)

Role	Sessions	Primary Topics Covered
CTO	3 sessions (architecture, strategy, diligence priorities)	Overall architecture direction, Series C requirements, team capacity, decision authority
VP Engineering	3 sessions (architecture, team, delivery)	Engineering team structure, delivery velocity, technical debt backlog, Phase 1 sequencing
Head of Platform	2 sessions (infrastructure, operations)	AWS account architecture, Terraform coverage, SRE practices, incident management
Senior Staff Engineer (Application Platform)	2 sessions	Rails application architecture, multi-tenancy implementation, database schema, notification service
Senior Staff Engineer (IoT Path)	2 sessions	MQTT broker configuration, ingestion endpoint design, telemetry volume patterns, device authentication
Head of Security	1 session	SOC 2 status, penetration test history, multi-tenant isolation review findings, Series C security diligence prep
Head of Hardware Operations	2 sessions	Device fleet management, firmware update processes, connectivity patterns, device failure modes
SRE 1	2 sessions (incidents, observability)	Incident post-mortems, CloudWatch coverage gaps, on-call burden, deployment processes
SRE 2	1 session (infrastructure)	EC2 sizing, RDS configuration, Auto Scaling behaviour, Redis usage patterns
VP Product	1 session (product roadmap)	AI capability requirements, customer-facing feature roadmap, Series C product narrative

B. Framework References

- AWS Well-Architected Framework (2024). Five pillars applied: Operational Excellence, Security, Reliability, Performance Efficiency, Cost Optimization. IoT Lens addendum applied to device and ingestion path assessment.
- AWS IoT Reference Architecture (2024). Message broker, device management, and telemetry processing patterns referenced for IoT Core and Kafka integration design.
- NIST Cybersecurity Framework (CSF) 2.0. Applied to threat model structure and SOC 2 control gap analysis in Section 10.
- DORA 2024 State of DevOps Report. Deployment frequency and change failure rate benchmarks referenced for Phase 1 operational maturity targets.
- TimescaleDB Documentation: Hypertables, Continuous Aggregates, and Compression (2024). Chunk sizing, compression ratio, and continuous aggregate refresh patterns applied to the time-series storage design.
- Kafka: The Definitive Guide (Narkhede, Shapira, Palino, 2nd ed.). Partition strategy, consumer group isolation, and log compaction patterns referenced for the streaming layer design.
- SOC 2 Trust Services Criteria (AICPA, 2022). CC6 (Logical and Physical Access) and CC7 (System Operations) control categories specifically referenced in Section 10.
- Clixlogix Consulting Practice IoT-SaaS Platform Modernization Framework (internal). Applied to assessment sequencing, evidence stream design, migration pattern selection, and phase gate criteria.

C. Glossary of Technical Terms

Term	Definition
Active-active (multi-region)	A deployment model where two or more geographic regions each serve production traffic simultaneously, providing both horizontal capacity and regional failover without a cold standby region
Aurora Global Database	An AWS-managed PostgreSQL-compatible database feature that replicates data across multiple AWS regions with sub-second lag and supports promotion of a secondary region to primary in under 30 seconds
Continuous aggregate	A TimescaleDB feature that automatically materialises pre-computed aggregations (e.g., per-minute averages) from raw time-series data, enabling fast time-range queries without scanning raw data
EKS (Elastic Kubernetes Service)	AWS's managed Kubernetes service, which handles the Kubernetes control plane and reduces the operational overhead of running containerized workloads
Feature flag	A software mechanism that allows a capability to be enabled or disabled at runtime without a deployment, enabling progressive rollout and instant rollback
Hypertable	The core TimescaleDB data structure: a PostgreSQL table that is automatically partitioned by time and optionally by a secondary dimension (in this case, building_id)
KEDA (Kubernetes Event-Driven Autoscaling)	A Kubernetes add-on that enables pod autoscaling based on event-driven metrics such as Kafka consumer lag, rather than only CPU and memory

Term	Definition
Log compaction (Kafka)	A Kafka topic configuration that retains only the most recent message for each unique key, providing a durable snapshot of the latest state for each key while discarding superseded versions
MSK (Managed Streaming for Apache Kafka)	AWS's managed Apache Kafka service, which handles broker provisioning, scaling, patching, and replication without requiring the operations team to manage Kafka infrastructure directly
MQTT (Message Queuing Telemetry Transport)	A lightweight publish-subscribe messaging protocol designed for constrained devices and unreliable networks; the standard IoT device-to-cloud communication protocol
OTA (Over-the-Air update)	A mechanism for delivering software or firmware updates to IoT devices remotely over their network connection, without requiring physical access
Partition key (Kafka)	A value used to determine which Kafka partition a message is written to. Messages with the same partition key are always written to the same partition, ensuring ordered delivery and enabling stateful processing
Row-level security (RLS)	A PostgreSQL feature that enforces per-row access policies at the database engine level, ensuring that a query only returns rows the authenticated user or application context is authorised to see
SPIFFE/SPIRE	A standard (SPIFFE: Secure Production Identity Framework For Everyone) and its reference implementation (SPIRE) for issuing verifiable cryptographic identity to software workloads, enabling service-to-service mutual authentication
TimescaleDB	An open-source time-series database built as an extension to PostgreSQL, providing automatic time-based partitioning, compression, and continuous aggregates for high-volume time-series data while remaining fully SQL-compatible
X.509 certificate	A digital certificate standard used for device authentication in this context; each IoT device carries a unique X.509 certificate that proves its identity to AWS IoT Core without requiring a username or password
Zero-trust security model	A security architecture principle that assumes no network location or component is inherently trusted; every request, service call, or database connection must carry a verifiable identity and be authorised explicitly

End of Report

Clixlogix Consulting Practice | info@clixlogix.com | clixlogix.com