David McNabb

david@aiapplied.uk

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Document Title

Subtitle

Smart Manufacturing Data Hub (SMDH)

**Architecture Design Options**

Comprehensive Analysis of Four Architecture Approaches

|  |  |
| --- | --- |
| Document Version | 2.0 FINAL |
| Date | October 2025 |
| Status | For Review |
| Classification | Internal Use |
| Author | AI Applied Technical Team |
| Intended Audience | Technical Architects and Business Stakeholders |

# 1. Executive Summary

This document presents a comprehensive analysis of four distinct architectural approaches for the Smart Manufacturing Data Hub (SMDH) platform. The SMDH is designed as a cloud-native, multi-tenant Internet of Things (IoT) platform that empowers small and medium-sized manufacturing enterprises with real-time visibility into their operations.

## 1.1 Document Purpose

The purpose of this document is to provide both technical architects and business stakeholders with a clear understanding of four viable architecture options for the SMDH platform. Each option has been analysed in detail, considering factors such as cost implications, technical complexity, performance characteristics, scalability, implementation timelines, and operational considerations.

## 1.2 The Four Architecture Options at a Glance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Option A: Flink | Option B: Snowflake | Option C: SiteWise | Option D: Timestream |
| Monthly Cost (30 tenants) | £2,500-4,200 | £2,170-3,450 | £6,334 | £3,965 |
| Complexity | Very High | Low | Medium-High | Medium |
| Timeline | 24 weeks | 20 weeks | 28 weeks | 22 weeks |
| Best Suited For | Sub-second latency required | Most scenarios (Recommended) | SiteWise expertise available | AWS-native mandate |

## 1.3 Key Finding: All Options Meet Budget Requirements

An important finding from our analysis is that all four options meet the budget constraint of £200-300 per tenant per month. A previous cost calculation error for Option C has been corrected, reducing its estimated monthly cost by 70% (from £21,150 to £6,334 for 30 tenants). This means the decision should be based on factors such as technical requirements, team capabilities, and strategic preferences rather than basic affordability.

## 1.4 Recommended Approach

**🏆 Recommended: Option B (Snowflake-Leveraged Architecture)**

For the majority of manufacturing IoT scenarios, Option B (Snowflake) provides the best balance of:

* Lowest cost: £2,170-3,450/month (£72-115 per tenant)
* Lowest complexity: Only 5 core services to manage
* Fastest time-to-market: 20 weeks to production
* SQL-first development: Familiar to most engineering teams
* Native multi-tenancy: Secure row-level security built-in
* Proven in manufacturing: Used by major industrial companies

The default configuration provides 60-65 second dashboard update latency, which can be improved to <5 seconds for critical alerts by adding a Lambda "fast-path" (additional £100-150/month and 2 weeks implementation).

## 1.5 Alternative Options

Whilst Option B is recommended as the default choice, the other options may be preferable in specific circumstances:

**Option D (AWS-Native):** Select this if your organisation has a mandate for AWS-only services (no Snowflake). Provides excellent multi-tenancy, Grafana dashboards, and unified time-series storage. Cost: £3,965/month (15% premium over Option B).

**Option A (Flink-Based):** Select this ONLY if sub-second latency is legally or regulatorily required for all data processing. Provides advanced stream processing but requires Flink expertise and accepts highest complexity. Cost: £2,500-4,200/month.

**Option C (SiteWise):** Select this only if your team has deep AWS IoT SiteWise expertise and you're serving <10 tenants (multi-tenancy limitations). Provides managed asset modelling but requires custom dashboard development. Cost: £6,334/month (most expensive).

## 1.6 How to Use This Document

This document is structured to support both detailed technical review and high-level business decision-making:

* Technical Architects: Review Sections 4-7 for detailed component descriptions, data flows, and technical trade-offs for each option
* Business Stakeholders: Focus on Section 1 (this summary), Section 8 (comparisons), and Section 9 (recommendations)
* Decision-Makers: Review the comparison matrices in Section 8 and the decision framework in Section 8.5
* All Readers: Refer to Section 10 (Glossary) for explanations of technical terms. Terms are explained in plain English throughout the document.

Each architecture option (Sections 4-7) includes a placeholder for inserting the corresponding architecture diagram created in Draw.io. Please insert these diagrams before final distribution.

Table of Contents

[1 1. Executive Summary 2](#_Toc212624864)

[1.1 1.1 Document Purpose 2](#_Toc212624865)

[1.2 1.2 The Four Architecture Options at a Glance 2](#_Toc212624866)

[1.3 1.3 Key Finding: All Options Meet Budget Requirements 2](#_Toc212624867)

[1.4 1.4 Recommended Approach 2](#_Toc212624868)

[1.5 1.5 Alternative Options 3](#_Toc212624869)

[1.6 1.6 How to Use This Document 3](#_Toc212624870)

[2 Table of Contents 4](#_Toc212624871)

[4 System Overview and Requirements 5](#_Toc212624872)

[4.1 What is the Smart Manufacturing Data Hub? 5](#_Toc212624873)

[4.1.1 How It Works (Simplified) 5](#_Toc212624874)

[4.2 Key System Requirements 5](#_Toc212624875)

[4.2.1 Data Volume and Performance Requirements 5](#_Toc212624876)

[4.2.2 Multi-Tenancy and Data Isolation 6](#_Toc212624877)

[4.2.3 Budget Constraints 7](#_Toc212624878)

[4.3 Use Cases Supported 7](#_Toc212624879)

[4.3.1 Use Case 1: Machine Utilisation Monitoring 7](#_Toc212624880)

[4.3.2 Use Case 2: Air Quality Management 8](#_Toc212624881)

[4.3.3 Use Case 3: Energy Monitoring and Optimisation 9](#_Toc212624882)

[4.3.4 Use Case 4: Job Location Tracking 10](#_Toc212624883)

[5 Option A: Flink-Based AWS-Heavy Architecture 11](#_Toc212624884)

[5.1 Overview and Summary 11](#_Toc212624885)

[5.2 Architecture Diagram 11](#_Toc212624886)

[5.3 3.3 Core Components 12](#_Toc212624887)

[5.3.1 Ingestion Layer 12](#_Toc212624888)

[5.3.2 Stream Processing Layer (Apache Flink) 12](#_Toc212624889)

[5.3.3 Data Platform Layer (Snowflake) 13](#_Toc212624890)

[5.3.4 Application Layer 13](#_Toc212624891)

[5.4 Data Flow 13](#_Toc212624892)

[5.5 Strengths 14](#_Toc212624893)

[5.6 Limitations 14](#_Toc212624894)

[5.7 When to Choose 14](#_Toc212624895)

[6 Architecture Option B: Snowflake-Leveraged Architecture 15](#_Toc212624896)

[6.1 Overview and Summary 15](#_Toc212624897)

[6.2 4.2 Architecture Diagram 15](#_Toc212624898)

[6.3 4.3 Core Components 15](#_Toc212624899)

[6.3.1 Simplified Ingestion 15](#_Toc212624900)

[6.3.2 Unified Data Platform (Snowflake Does It All) 15](#_Toc212624901)

[6.3.3 Machine Learning (Snowflake Cortex) 16](#_Toc212624902)

[6.4 Data Flow 16](#_Toc212624903)

[6.5 Strengths 17](#_Toc212624904)

[6.6 Limitations 17](#_Toc212624905)

[6.7 When to Choose 17](#_Toc212624906)

[7 Architecture Option C: AWS IoT SiteWise-Based 17](#_Toc212624907)

[7.1 Overview 17](#_Toc212624908)

[7.2 Architecture Diagram 18](#_Toc212624909)

[8 Option D: AWS-Native (Timestream + Grafana) 19](#_Toc212624910)

[8.1 Overview 19](#_Toc212624911)

[8.2 6.2 Architecture Diagram 19](#_Toc212624912)

[9 7. Architecture Comparison 20](#_Toc212624913)

[9.1 7.1 Cost Comparison 20](#_Toc212624914)

[9.2 7.2 Complexity Comparison 20](#_Toc212624915)

[9.3 7.3 Performance and Latency 20](#_Toc212624916)

[9.4 7.4 Multi-Tenancy Security 20](#_Toc212624917)

[9.5 7.5 Decision Framework 21](#_Toc212624918)

# System Overview and Requirements

## What is the Smart Manufacturing Data Hub?

The Smart Manufacturing Data Hub (SMDH) is a cloud-based software platform designed to help small and medium-sized manufacturing enterprises (SMEs) monitor and analyse their operations in real-time. This centralised "command centre" where data from factory equipment, environmental sensors, energy metres, and tracking systems flows together to provide actionable insights.

Unlike traditional manufacturing systems that require extensive IT infrastructure and expertise, SMDH is designed as a self-service platform. Companies can independently register, configure their equipment, and start viewing data—all through an intuitive web interface without requiring specialised technical knowledge.

### How It Works (Simplified)

The platform operates through a straightforward workflow:

**1. Company Registration:** Manufacturing companies create accounts through a self-service portal. They provide basic information about their organisation and can immediately begin setup.

**2. Site and Device Configuration:** Companies register their manufacturing facilities (sites) and the equipment they want to monitor. A guided wizard helps configure sensors, metres, and tracking devices with minimal technical input.

**3. Automatic Data Collection:** Once configured, sensors begin sending data automatically. Machine status, environmental readings, energy consumption, and job locations flow continuously into the platform.

**4. Data Processing and Storage:** The system receives, validates, and stores data securely. Each company's data is completely isolated from others. Historical data is retained according to compliance requirements.

**5. Real-Time Dashboards:** Pre-built dashboards automatically display relevant information based on the types of equipment registered. Users see real-time status, historical trends, and analytics.

**6. Alerts and Notifications:** The system monitors data against configurable thresholds. When issues arise (equipment failure, poor air quality, excessive energy use), users receive immediate notifications via email or SMS.

## Key System Requirements

### Data Volume and Performance Requirements

The platform must handle significant data volumes whilst maintaining responsive performance:

|  |  |
| --- | --- |
| Requirement | Target Specification |
| Daily data ingestion | 2.6 to 3.9 million data points per day |
| Real-time monitoring latency | Under 1 second for dashboard updates |
| Alert notification time | Under 10 seconds from trigger event to notification |
| Dashboard query response | Under 5 seconds for complex analytics queries |
| System availability | 99.9% uptime (maximum 8.76 hours downtime per year) |

These requirements ensure users have immediate visibility into their operations. A machine failure or air quality issue must be detected and escalated within seconds, not minutes, to enable timely response.

### Multi-Tenancy and Data Isolation

As a Software-as-a-Service (SaaS) platform serving multiple manufacturing companies simultaneously, the SMDH must provide absolute data isolation. This is non-negotiable for several reasons:

* Security: Company A must never access Company B's data under any circumstances
* Compliance: GDPR and industry regulations require strict data segregation
* Intellectual Property: Manufacturing data often contains trade secrets and proprietary processes
* Performance: One company's heavy usage must not impact another company's performance
* Scalability: The system must efficiently serve 30-40 companies initially, scaling to 100+ over time

**Technical Implementation:**

Multi-tenancy can be implemented in several ways, and this is a key differentiator between the four architecture options:

**• Row-Level Security (RLS):** Database-enforced filtering that automatically restricts queries to authorised data. Used in Options A and B with Snowflake. Most secure.

**• Partition Keys:** Physical data separation using tenant identifiers as partition keys. Used in Option D with Timestream. Good security with performance benefits.

**• Tag-Based Filtering:** Application-layer filtering using metadata tags. Used in Option C with SiteWise. Requires careful implementation to avoid data leakage.

### Budget Constraints

The platform has a clear budget target to ensure economic viability for SMEs:

Target Cost per Tenant: £200-300 per month.

For an initial deployment of 30 manufacturing companies, this translates to:

|  |  |
| --- | --- |
| Budget Component | Amount |
| Total Monthly Infrastructure Cost | £6,000 - £9,000 (30 tenants) |
| Per-Tenant Cost Target | £200 - £300 |
| Annual Infrastructure Budget | £72,000 - £108,000 |

All four architecture options evaluated in this document meet this budget constraint. The decision comes down to factors other than basic affordability, such as complexity, team capabilities, and specific technical requirements.

## Use Cases Supported

The SMDH platform must support diverse monitoring scenarios across manufacturing environments. Each use case has distinct characteristics and requirements:

### Use Case 1: Machine Utilisation Monitoring

**Business Objective:** Track how efficiently manufacturing equipment is being used to identify bottlenecks, reduce downtime, and improve overall equipment effectiveness (OEE).

**How It Works:**

Sensors attached to manufacturing equipment (CNC machines, lathes, presses, mills) collect data every second about the machine's operational state:

* State: Running, Idle, or Offline
* Cycle count: Number of production cycles completed
* Performance: Actual vs expected production rate
* Downtime events: When and why machines stop
* Error codes: Specific failure conditions

**Technical Specifications:**

|  |  |
| --- | --- |
| Specification | Value |
| Data frequency | 1 Hz (one reading per second) |
| Sensors per site | 30-45 machines typically |
| Communication protocol | LoRaWAN or MQTT |
| Daily data volume | ~2.6 million readings per site per day |
| Latency requirement | <1 second (immediate status visibility required) |

**Key Metrics Calculated:**

* Overall Equipment Effectiveness (OEE): Industry-standard metric combining availability, performance, and quality (target: >85% world-class)
* Availability: Percentage of scheduled time the machine is operational
* Performance: Actual production rate vs ideal production rate
* Utilisation: Percentage of time machines are actively producing
* Mean Time Between Failures (MTBF): Reliability metric
* Mean Time To Repair (MTTR): Maintenance efficiency metric

### Use Case 2: Air Quality Management

**Business Objective:** Monitor environmental conditions in manufacturing facilities to ensure worker health and safety, regulatory compliance, and optimal working conditions.

**How It Works:**

Environmental sensors positioned throughout the facility continuously monitor air quality parameters:

* CO₂ levels: Carbon dioxide concentration (target: <1000 ppm for good ventilation)
* VOCs: Volatile Organic Compounds from paints, solvents, adhesives
* Particulate Matter: PM1, PM2.5, PM4, PM10 (from grinding, cutting, welding)
* Temperature: Workspace thermal comfort
* Humidity: Moisture levels affecting comfort and processes
* Atmospheric Pressure: Baseline environmental measurement

**Technical Specifications:**

|  |  |
| --- | --- |
| Specification | Value |
| Data frequency | 1-minute intervals |
| Sensors per site | 10-15 sensors (distributed across facility) |
| Communication protocol | MQTT over WiFi/Ethernet |
| Daily data volume | ~200,000 readings per site per day |
| Alert requirement | <10 seconds for dangerous conditions (e.g., CO₂ >5000 ppm) |

**Regulatory Compliance:**

The platform must support compliance with:

* UK Health and Safety Executive (HSE) Workplace Exposure Limits (WELs)
* European Union Occupational Safety and Health Directives
* ISO 45001 Occupational Health and Safety Management
* COSHH (Control of Substances Hazardous to Health) Regulations

### Use Case 3: Energy Monitoring and Optimisation

**Business Objective:** Track electrical consumption to identify energy waste, reduce costs, and meet sustainability goals. Manufacturing typically represents 30-50% of operational costs for energy-intensive facilities.

**How It Works:**

Energy monitoring devices (installed at circuit breaker panels or on individual equipment) measure electrical parameters in real-time:

* Voltage: Electrical potential (V) - indicates power quality
* Current: Electrical flow (A) - indicates load
* Power Factor: Efficiency of electricity usage (target: >0.95)
* Real Power: Actual energy consumed (kW)
* Apparent Power: Total power drawn (kVA)
* Cumulative Consumption: Total kilowatt-hours (kWh) over time
* Cost Estimate: Energy cost based on tariff rates

**Technical Specifications:**

|  |  |
| --- | --- |
| Specification | Value |
| Data frequency | 15-second intervals |
| Monitors per site | 10-20 monitoring points (circuits + individual equipment) |
| Communication protocol | Modbus TCP or MQTT |
| Daily data volume | ~600,000 readings per site per day |
| Accuracy requirement | ±1% for billing-grade monitoring |

**Key Analytics:**

* Baseline Consumption: Establish normal usage patterns
* Peak Demand Analysis: Identify when and where peak usage occurs (affects tariffs)
* Power Factor Correction Opportunities: Improve efficiency and reduce reactive power charges
* Equipment Efficiency Comparison: Compare energy use across similar machines
* Cost Allocation: Distribute energy costs across departments or products
* Carbon Footprint Calculation: Convert kWh to CO₂ emissions for sustainability reporting

### Use Case 4: Job Location Tracking

**Business Objective:** Provide real-time visibility of work-in-progress (WIP) locations throughout the factory floor. Prevents lost jobs, reduces search time, and enables accurate delivery commitments.

**How It Works:**

Jobs are tagged with RFID tags or printed barcodes. As jobs move through the manufacturing process, workers or automated scanners record each movement:

* Job Start: When work begins on an order
* Station Arrival: Job reaches a workstation (e.g., "Welding Station 3")
* Station Completion: Work at that station finishes
* Quality Check: Inspection points
* Job Completion: Final product ready for shipping
* Exception Events: Holds, rework required, quality failures

**Technical Specifications:**

|  |  |
| --- | --- |
| Specification | Value |
| Event type | Discrete events (not continuous time-series) |
| Event frequency | 500-2,000 scans per site per day |
| Scanners per site | 5-15 RFID readers or barcode scanners |
| Communication protocol | HTTP REST API or MQTT |
| Data structure | Event-based with rich metadata (job ID, location, operator, timestamp) |

**Key Capabilities:**

* Real-Time Location: "Where is Job #12345 right now?"
* Job History: Complete audit trail of movements
* Dwell Time Analysis: How long jobs spend at each station (identifies bottlenecks)
* Exception Alerting: Jobs stuck at one location beyond expected time
* Delivery Estimates: Predict completion time based on current location and historical data
* Throughput Metrics: Jobs completed per hour/day by station

**Important Architectural Consideration:**

Job tracking events have fundamentally different characteristics than continuous sensor data. They are discrete, irregular, and event-driven rather than time-series. This impacts architecture choice:

* Options A, B, D: Handle job tracking naturally alongside time-series data
* Option C (SiteWise): SiteWise is designed for continuous time-series, not discrete events. Requires separate Timestream database, increasing complexity.

# Option A: Flink-Based AWS-Heavy Architecture

## Overview and Summary

|  |  |
| --- | --- |
| Characteristic | Details |
| Monthly Cost (30 tenants) | £2,500 - 4,200 |
| Cost Per Tenant | £83 - 140 |
| Complexity Level | Very High (15+ AWS services) |
| Implementation Timeline | 24 weeks |
| Team Size & Skills | 3-4 engineers (Flink, Java/Scala, AWS, data engineering) |
| Best Suited For | Sub-second latency legally required; advanced ML needed |

Option A represents the most technically sophisticated architecture, leveraging Apache Flink for advanced stream processing. This approach provides unparalleled real-time capabilities with sub-second latency but comes with significant operational complexity.

## Architecture Diagram

*[INSERT DIAGRAM: SMDH-Option-A-Flink-Architecture.drawio]*

## 3.3 Core Components

### Ingestion Layer

**AWS IoT Core:** MQTT broker receiving 1Hz sensor data from 30-45 machines per site. Handles device authentication via X.509 certificates. Provides device shadows for configuration.

**IoT Rules Engine:** Routes messages based on content. Enriches data with tenant metadata. Filters invalid messages. SQL-based routing rules.

**Kinesis Data Firehose:** Buffers data in 10MB or 60-second batches. Compresses to Parquet format. Invokes Lambda for transformation. Delivers to S3.

**Lambda Transformation:** Validates sensor ranges, converts units, enriches with lookup data. Python or Node.js. Scales automatically.

**S3 Data Lake:** Stores Parquet files partitioned by tenant/date. Lifecycle policies for archival. Enables historical analysis and recovery.

### Stream Processing Layer (Apache Flink)

**Why Flink?**

Apache Flink is an advanced distributed stream processing framework. Unlike simpler alternatives, Flink provides:

* Sub-second latency: Process and respond to events in <1 second
* Exactly-once semantics: Guarantees no data loss or duplication
* Complex event processing: Detect patterns across multiple streams
* Stateful processing: Remember context across events (e.g., calculate 5-minute rolling averages)
* Advanced windowing: Tumbling, sliding, session windows for aggregations
* Backpressure handling: Gracefully manages data spikes

**Flink on EMR:** Runs on 3-5 node cluster. Auto-scaling based on data velocity. Checkpoints state to S3 every 5 minutes.

**RocksDB State Backend:** Embedded database storing intermediate state. Enables fault tolerance. Recovers from checkpoints if nodes fail.

**Flink Jobs:** Custom Java/Scala code for: real-time aggregations, anomaly detection, pattern matching, multi-tenant data isolation.

### Data Platform Layer (Snowflake)

**Snowpipe Auto-Ingestion:** Detects new S3 files via events. Loads into Snowflake within seconds. Serverless—no warehouse management.

**Raw Tables:** VARIANT columns store semi-structured JSON. Partitioned by tenant\_id. Clustered on timestamp for query performance.

**Snowflake Streams:** Change Data Capture tracking new/changed rows. Enables incremental processing. Multiple consumers can read independently.

**Snowflake Tasks:** Scheduled SQL transformations. DAG-based dependencies. Runs hourly aggregations, data quality checks, archival.

**Curated Views:** Row-Level Security enforces tenant isolation. Pre-calculated aggregations. Analytics-ready datasets.

### Application Layer

**API Gateway + Lambda:** REST APIs for data access. Tenant authentication via Cognito. Query Snowflake and return JSON.

**ECS Fargate Web App:** React frontend hosted on Fargate. Auto-scaling. CI/CD via CodePipeline.

**QuickSight Dashboards:** Embedded analytics. Row-Level Security passes through from Snowflake. Pay-per-session pricing.

**SageMaker ML:** Custom model training (predictive maintenance, anomaly detection). GPU-accelerated. Model hosting on auto-scaling endpoints.

## Data Flow

Real-time path (critical alerts):

1. Sensor → IoT Core (100ms)
2. IoT Rules → Kinesis Streams (200ms)
3. Flink processes and detects alert condition (500ms)
4. SNS notification sent (200ms)
5. Total: <2 seconds end-to-end

Batch path (historical analytics):

1. Kinesis Firehose buffers 60 seconds
2. Lambda transforms batch
3. S3 receives Parquet files
4. Snowpipe loads into Snowflake (<5 min)
5. Snowflake Tasks aggregate hourly
6. QuickSight dashboards refresh

## Strengths

**Sub-Second Latency:** Achieves <1s for critical data paths. Essential if regulatory requirements mandate immediate response.

**Advanced Stream Processing:** Complex event processing, pattern detection, sophisticated windowing not possible with simpler systems.

**Unlimited Flexibility:** Custom Java/Scala code enables any transformation or ML algorithm.

**Proven at Scale:** Flink powers streaming at Netflix, Uber, Alibaba processing billions of events daily.

## Limitations

**Very High Complexity:** Managing 15+ services, Flink cluster tuning, distributed debugging.

**Flink Expertise Required:** Rare and expensive skill set. Steep learning curve for Java/Scala.

**Longest Timeline:** 24 weeks to production. Complex distributed systems take time to build and test.

**Operational Overhead:** EMR cluster management, checkpointing monitoring, state recovery testing.

## When to Choose

Select Option A ONLY if:

* Sub-second latency is legally or regulatorily mandated (not just preferred)
* Complex event processing is essential for the business case
* Team has Flink expertise or budget for specialists
* Willing to accept highest complexity and 24-week timeline

# Architecture Option B: Snowflake-Leveraged Architecture

## Overview and Summary

|  |  |
| --- | --- |
| Characteristic | Details |
| Monthly Cost (30 tenants) | £2,170 - 3,450 (CHEAPEST) |
| Cost Per Tenant | £72 - 115 |
| Complexity Level | Low (5 core services) |
| Implementation Timeline | 20 weeks (FASTEST) |
| Team Size & Skills | 2-3 engineers (SQL, Snowflake, basic AWS) |
| Best Suited For | Most scenarios - default recommendation |

Option B centralises data processing in Snowflake, simplifying the architecture compared to Option A. Leveraging Snowflake's native streaming, task orchestration, and SQL-first approach, this option provides the fastest time-to-market and lowest operational complexity whilst meeting all functional requirements.

## 4.2 Architecture Diagram

*[INSERT DIAGRAM: SMDH-Option-B-Snowflake-Architecture.drawio]*

## 4.3 Core Components

### Simplified Ingestion

**AWS IoT Core:** Same as Option A—MQTT broker with X.509 authentication.

**Kinesis Data Streams:** Lightweight buffering (not Firehose). Streams directly to Snowflake. 10-20 shards partitioned by tenant\_id.

**Snowpipe Streaming:** Snowflake's streaming ingestion service. Loads data in 5-10 seconds. Eliminates S3 staging and Lambda transformation complexity.

### Unified Data Platform (Snowflake Does It All)

**Key Simplification:**

In Option B, Snowflake replaces multiple Option A services:

* Replaces S3 Data Lake → Snowflake internal storage
* Replaces Lambda + Flink → Snowflake Streams and Tasks
* Replaces AWS Glue → Snowflake native schema evolution
* Replaces separate ETL orchestration → Snowflake Task DAGs

**Raw Tables with VARIANT:** JSON data stored in VARIANT columns. Schema-flexible. Automatic compression (75-85% reduction). Partitioned by tenant\_id.

**Snowflake Streams (CDC):** Tracks inserts/updates/deletes. Enables incremental processing. Zero-copy—doesn't duplicate data.

**Snowflake Tasks:** SQL-based transformations run on schedule or when streams have data. DAG orchestration. Serverless compute.

**Dynamic Tables:** Continuously updated aggregations. Declarative SQL definitions. Incremental refresh automatically.

**Row-Level Security (RLS):** Native multi-tenancy. Policies enforce tenant isolation at database level. Cannot be bypassed by applications.

### Machine Learning (Snowflake Cortex)

Snowflake Cortex ML enables SQL-based machine learning:

* CREATE MODEL predict\_failure AS SELECT \* FROM training\_data;
* SELECT PREDICT\_FAILURE(sensor\_data) FROM current\_readings;
* Supports: Classification, regression, time-series forecasting
* Limitation: Less flexible than SageMaker but covers 80% of use cases

## Data Flow

End-to-end flow (simplified):

1. Sensor → IoT Core (100ms)
2. Kinesis Streams (200ms)
3. Snowpipe Streaming → Raw table (5-10 seconds)
4. Snowflake Stream detects new rows immediately
5. Task processes (runs every 1 min) → Curated views
6. Dashboard refreshes (total: 60-65 seconds)

**Alert Latency Solution:**

Default 60s latency violates <10s alert requirement. FIX: Add Lambda fast-path for alerts (+£100-150/month, +2 weeks). Fast-path achieves <5s alert latency whilst keeping batch processing in Snowflake.

## Strengths

**Lowest Cost:** £2,170-3,450/month. Cheapest option by 15-20%. Auto-suspend when idle.

**Lowest Complexity:** Only 5 core services vs 15+ in Option A. Single platform for most processing.

**Fastest Timeline:** 20 weeks to production. SQL-first development accelerates delivery.

**SQL-First Development:** Familiar to most engineers. Easier to hire than Flink specialists.

**Native Multi-Tenancy:** Row-Level Security enforced at database level. Production-proven.

**Unified Governance:** All data in one platform. Single place for audits, lineage, retention policies.

## Limitations

**Alert Latency Requires Fix:** 60s default. Add Lambda fast-path for <5s alerts (+£100-150/month).

**Snowflake Licensing:** Separate from AWS bill.

**Not Sub-Second:** Snowpipe Streaming is 5-10s, not <1s. If sub-second legally required, choose Option A.

## When to Choose

Select Option B if:

* Cost and simplicity are priorities (most SME scenarios)
* Team has SQL skills or can train quickly
* 60s dashboard latency acceptable, or willing to add Lambda fast-path
* Want fastest time to market (20 weeks)
* Snowflake licensing not a blocker

# Option C: AWS IoT SiteWise-Based

|  |  |
| --- | --- |
| Monthly Cost | £6,334 (CORRECTED) |
| Complexity | Medium-High (8-10 services) |
| Timeline | 28 weeks |
| Best For | SiteWise expertise; <10 tenants |
| Key Limitation | NOT designed for multi-tenant SaaS |

|  |  |
| --- | --- |
| Monthly Cost | £3,965 |
| Complexity | Medium (7-8 services) |
| Timeline | 22 weeks |
| Best For | AWS-native mandate |
| Key Strength | Fully AWS-native, Grafana dashboards |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Option | Monthly (30 tenants) | Per Tenant | Annual TCO | Within Budget? | Rank |
| B (Snowflake) | £2,170-3,450 | £72-115 | £346K-521K | ✅ Yes | 1st (Cheapest) |
| A (Flink) | £2,500-4,200 | £83-140 | £525K-720K | ✅ Yes | 2nd |
| D (Timestream) | £3,965 | £132 | £374K-538K | ✅ Yes | 3rd |
| C (SiteWise) | £6,334 | £211 | £566K-715K | ✅ Yes (corrected) | 4th (Most expensive) |

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Number of Services | Primary Skills Required | Complexity Rank |
| B (Snowflake) | 5 services | SQL, Snowflake | 1st (Simplest) |
| D (Timestream) | 7-8 services | AWS-native, SQL, Grafana | 2nd |
| C (SiteWise) | 8-10 services | SiteWise, AWS IoT, React | 3rd |
| A (Flink) | 15+ services | Flink, Java/Scala, AWS | 4th (Most complex) |

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Real-Time Latency | Alert Latency | Dashboard Updates |
| A (Flink) | <1 second | <5 seconds ✅ | <5 seconds |
| B (Snowflake) | 5-10 seconds | 60s (fixable to <5s) | 60-65 seconds |
| C (SiteWise) | <1 second | <10 seconds ✅ | <5 seconds |
| D (Timestream) | <5 seconds | <5 seconds ✅ | <10 seconds |

|  |  |  |
| --- | --- | --- |
| Option | Multi-Tenancy Approach | Security Assessment |
| B (Snowflake) | Row-Level Security (RLS) | ✅ Excellent - Database enforced |
| A (Flink) | Snowflake RLS | ✅ Excellent - Database enforced |
| D (Timestream) | Partition Keys | ✅ Good - Native physical isolation |
| C (SiteWise) | Tag-based filtering | ⚠️ Risky - Application-layer |

## 5.1 Overview and Summary

|  |  |
| --- | --- |
| Characteristic | Details |
| Monthly Cost (30 tenants) | £6,334 ✅ CORRECTED (was £21,150) |
| Cost Per Tenant | £211 |
| Complexity Level | Medium-High (8-10 AWS services + dual storage) |
| Implementation Timeline | 28 weeks (longest - custom dashboards required) |
| Team Size & Skills | 3-4 engineers (SiteWise, AWS IoT, React, Timestream) |
| Best Suited For | Teams with deep SiteWise expertise; <10 tenants; AWS-native preference |

**⚠️ IMPORTANT COST CORRECTION**

A previous analysis incorrectly calculated Option C cost at £21,150/month due to a 7.3x data volume error. The CORRECTED cost is £6,334/month (70% reduction), making this option economically viable. However, it remains the most expensive option and has multi-tenancy limitations for SaaS platforms.

Option C leverages AWS IoT SiteWise, a managed service purpose-built for industrial equipment monitoring. SiteWise provides native concepts like asset hierarchies (Factory → Line → Machine), automatic time-series calculations, and OPC-UA/Modbus protocol support. However, it is NOT designed for multi-tenant SaaS, requiring significant customisation and dual-storage architecture.

## 5.2 Architecture Diagram

*[INSERT DIAGRAM: SMDH-Option-C-SiteWise-Architecture.drawio]*

## 5.3 Core Components

### Dual Ingestion Path (SiteWise + Timestream)

**Why Dual Storage?**

Option C requires TWO separate data stores because SiteWise and Timestream serve different purposes:

|  |  |  |
| --- | --- | --- |
| Aspect | SiteWise | Timestream |
| Data Type | Continuous time-series (machine status, sensors) | Discrete events (RFID scans, job tracking) |
| Best For | Regular sensor readings, asset hierarchies, OEE | Irregular events, rich metadata, SQL queries |
| Use Cases | Machine utilisation, energy, air quality | Job location tracking, barcode scans |

**SiteWise Gateway (On-Premises):** Software appliance running at manufacturing sites. Translates industrial protocols (OPC-UA, Modbus TCP) to AWS IoT format. Handles local caching for offline operation. Requires management at each site (updates, security patches).

**AWS IoT Core:** MQTT broker (same as other options). Routes data based on type: continuous data → SiteWise, events → Kinesis.

**IoT Rules Engine:** Routes messages by content. Continuous sensor data goes to SiteWise, discrete events (RFID/barcodes) route to Kinesis Streams.

**Kinesis Streams (Events Only):** Buffers discrete events for ingestion into Timestream. Does NOT handle continuous sensor data (that goes directly to SiteWise).

### AWS IoT SiteWise (Time-Series Platform)

**What Makes SiteWise Different?**

Unlike general-purpose databases, SiteWise is purpose-built for industrial IoT:

* Asset Models: Templates defining equipment structure (e.g., "CNC Machine" with properties for spindle speed, temperature, state)
* Asset Hierarchies: Organise equipment in trees (Company → Site → Production Line → Machine → Sensor)
* Compute Expressions: Automatic calculations (OEE, availability, performance) without custom code
* Native Time-Series Storage: Optimised for high-frequency sensor data with automatic aggregations
* Built-in Formulas: Industry-standard metrics (OEE, MTBF, MTTR) pre-configured

**SiteWise Asset Ingestion:** Receives data from Gateway or direct MQTT. Validates against asset models. Stores in time-series database. Latency: <1 second.

**Asset Models:** Define structure: Measurements (raw sensor readings), Metrics (calculated values like OEE), Transforms (unit conversions, aggregations), Alarms (threshold-based alerts).

**Compute Expressions:** SiteWise Expression Language (similar to Excel formulas). Example: availability = (total\_time - downtime) / total\_time. Runs automatically as data arrives.

**Time-Series Store:** Hot tier: 13 months (fast queries). Cold tier: 14+ months (S3-based archival). Automatic aggregations: 1min, 5min, 1hour, 1day intervals.

**SiteWise Monitor:** Built-in dashboarding tool. LIMITATION: Cannot be adequately white-labelled for SaaS. AWS branding remains visible. Not multi-tenant aware.

### ⚠️ Critical Multi-Tenancy Challenge

**SiteWise is NOT Designed for Multi-Tenant SaaS**

Unlike Snowflake's Row-Level Security or Timestream's partition keys, SiteWise has NO native multi-tenancy support. The workaround:

* Tag every asset with "tenant\_id" metadata
* Application layer (Lambda/API) must filter ALL queries by tenant\_id
* Custom React application enforces filtering in UI code
* Requires extensive security testing—a bug could expose one tenant's data to another

**Risk Assessment:** Medium-high risk for multi-tenant SaaS. Acceptable for single-tenant deployments or <10 trusted tenants. Requires rigorous code review and penetration testing.

### Amazon Timestream (Event Data)

**Why Timestream?:** SiteWise cannot efficiently handle discrete events (RFID scans, job completions). These have irregular timing and rich metadata. Timestream is purpose-built for this.

**Timestream Tables:** Separate tables for job\_scan\_events, quality\_checks, maintenance\_logs. Partition key: tenant\_id (better multi-tenancy than SiteWise tags).

**SQL Query Engine:** Standard SQL interface for event analysis. Example: "Find all scans for Job #1234 in past week." Much simpler than SiteWise property queries.

### Custom Dashboard Layer (Required)

**Why Custom Dashboards Are Necessary**

SiteWise Monitor limitations force custom development:

* Cannot white-label: AWS branding and URLs remain visible
* Not multi-tenant: No concept of separating dashboards by company
* Limited customisation: Dashboard layout and chart types are constrained
* Cannot combine data: Cannot easily show SiteWise + Timestream data together

**Custom React Application:** Built from scratch using React 18+ and TypeScript. Calls SiteWise and Timestream APIs directly. Uses Chart.js or Recharts for visualisations. Fully white-labelable.

**API Gateway + Lambda:** Backend layer enforcing tenant isolation. Every API call validates user's tenant\_id before querying SiteWise/Timestream. Filters results by tenant.

**AWS Amplify Hosting:** Hosts React frontend. Provides CI/CD pipeline. Integrates with Cognito for authentication.

## 5.4 Data Flow

Time-series data path (Machine/Energy/Air Quality):

1. Sensor → OPC-UA/Modbus → SiteWise Gateway (on-prem)
2. Gateway → AWS IoT SiteWise ingestion (latency: <1s)
3. SiteWise validates against asset model, stores in time-series DB
4. Compute expressions calculate metrics (OEE, aggregations)
5. Custom React app queries SiteWise API for dashboard display (latency: <5s total)

Event data path (RFID/Barcode Scans):

1. Scanner → HTTP POST to API Gateway
2. Lambda validates and routes to Kinesis Streams (latency: <200ms)
3. Lambda consumer writes to Timestream (latency: <1s)
4. Custom React app queries Timestream SQL for job tracking dashboard (latency: <3s total)

## 5.5 Strengths

**Purpose-Built for Manufacturing:** Native industrial concepts: asset models, OEE, equipment hierarchies. Speaks manufacturing language, not generic database terms.

**Managed Asset Modelling:** Built-in templates for common equipment. Automatic calculations for availability, performance, quality. Less custom code than building from scratch.

**Real-Time Latency:** <1 second for time-series data. Meets real-time monitoring requirements without complex stream processing like Flink.

**Industrial Protocol Support:** SiteWise Gateway handles OPC-UA, Modbus TCP natively. Easier connectivity to legacy manufacturing equipment.

**Automatic Metric Calculations:** Compute expressions run at ingestion time. Standard manufacturing metrics (OEE, MTBF, utilisation) pre-configured.

## 5.6 Limitations

**NOT Designed for Multi-Tenant SaaS:** CRITICAL: No native Row-Level Security. Tag-based filtering is application-layer workaround with data leakage risk. Acceptable for <10 tenants; risky for SaaS.

**Most Expensive Option:** Cost: £6,334/month (£211/tenant) vs £2,170 for Option B (cheapest). 3x more expensive. Ingestion pricing ($0.75 per million values) adds up with 1Hz sensors.

**Dual Storage Complexity:** Requires BOTH SiteWise (time-series) AND Timestream (events). Applications must query two databases. Data correlation requires custom logic.

**Custom Dashboard Development:** SiteWise Monitor cannot be white-labelled. Must build custom React application (6-8 weeks development + ongoing maintenance).

**Longest Timeline:** 28 weeks to production. Custom dashboards, dual storage integration, and asset model configuration extend timeline 40% beyond Option B.

**On-Premises Gateway Management:** SiteWise Gateway requires software deployment at each manufacturing site. Software updates, troubleshooting, security patching across distributed locations.

**SiteWise Learning Curve:** Proprietary concepts (asset models, expression language) require training. Smaller talent pool than SQL or Python.

## 5.7 When to Choose

Select Option C ONLY if:

* Team has existing deep AWS IoT SiteWise expertise (rare)
* Manufacturing focus with <10 tenants (multi-tenancy risk acceptable)
* Already using SiteWise for other projects (leverage existing skills)
* OPC-UA/Modbus connectivity is critical and must be managed (not delegated)
* Budget supports £200+/tenant (most expensive option)
* Willing to invest 28 weeks + custom dashboard development
* Can accept dual storage complexity and manual multi-tenant isolation

**⚠️ ASSESSMENT:** Option C is viable after cost correction but NOT recommended for multi-tenant SaaS. The multi-tenancy risk (tag-based filtering), highest cost, dual storage complexity, and custom dashboard requirement outweigh the benefits of managed asset modelling. Choose Option B for simplicity or Option D for AWS-native without SiteWise limitations.

# 6. Architecture Option D: Pure AWS-Native Architecture (Timestream + Grafana)

## 6.1 Overview and Summary

|  |  |
| --- | --- |
| Characteristic | Details |
| Monthly Cost (30 tenants) | £3,965 |
| Cost Per Tenant | £132 |
| Complexity Level | Medium (7-8 AWS services) |
| Implementation Timeline | 22 weeks |
| Team Size & Skills | 2-3 engineers (AWS-native, SQL, Grafana) |
| Best Suited For | AWS-native mandate; organisations preferring no Snowflake |

**✅ BEST AWS-NATIVE OPTION**

Option D provides a fully AWS-native solution using Amazon Timestream for all time-series and event data, combined with Grafana for industry-leading dashboards. This approach avoids Snowflake licensing concerns (Option B) and SiteWise multi-tenancy limitations (Option C) whilst maintaining reasonable cost and complexity. It is the recommended choice when AWS-native services are mandated.

## 6.2 Architecture Diagram

*[INSERT DIAGRAM: SMDH-Option-D-Timestream-Architecture.drawio]*

## 6.3 Core Components

### Unified Ingestion (Single Data Path)

**Key Simplification vs Option C**

Unlike Option C which requires dual storage (SiteWise + Timestream), Option D uses Timestream for EVERYTHING—continuous sensor data AND discrete events. This dramatically simplifies the architecture.

* Single database to query (not two)
* Unified data model (no translation between SiteWise and Timestream)
* Single SQL interface (familiar to most engineers)
* Native partition-based multi-tenancy (better than SiteWise tags)
* Lower operational overhead (one database to monitor, backup, optimise)

**AWS IoT Core:** Same MQTT broker as other options. X.509 certificate authentication. Device shadows. QoS 0/1 support.

**IoT Rules Engine:** Routes ALL data types to Kinesis Streams. Simpler than Option C (no dual routing). Enriches messages with tenant metadata.

**Kinesis Data Streams:** 10-20 shards partitioned by tenant\_id. Buffers all sensor data and events. 24-hour retention for replay capability.

**Lambda Ingestion Handler:** Processes Kinesis batches. Validates sensor ranges. Converts to Timestream format. Writes to appropriate Timestream tables. Handles both time-series and events.

### Amazon Timestream (Unified Data Platform)

**Why Timestream Over SiteWise?**

|  |  |  |
| --- | --- | --- |
| Feature | SiteWise (Option C) | Timestream (Option D) |
| Cost Model | $0.75 per million values | $0.50 per GB ingested |
| Multi-Tenancy | Manual tags ⚠️ | Native partition keys ✅ |
| Event Data | Poor fit (needs Timestream) | Excellent (unified) ✅ |
| Query Language | Limited property APIs | Full SQL ✅ |
| Flexibility | Asset model constraints | Schemaless ✅ |
| Cost (30 tenants) | £6,334/month | £3,965/month ✅ |

**Verdict:** Timestream is 40% cheaper and more flexible for multi-tenant SaaS.

**Timestream Database Schema:** Separate tables for machine\_telemetry, energy\_monitoring, air\_quality, job\_scan\_events. ALL tables partitioned by tenant\_id (native physical isolation).

**Memory Store:** Hot tier: 7-90 days configurable. Fast queries for recent data. Used for real-time dashboards and alerts.

**Magnetic Store:** Warm/cold tier: Months to years retention. Cost-optimised storage. Used for historical analysis and compliance.

**SQL Query Engine:** Standard SQL (mostly ANSI-compatible). Familiar to most engineers. Supports JOINs, window functions, aggregations. Time-series functions (interpolate, binning).

**Scheduled Queries:** Native support for pre-aggregation. Runs SQL queries on schedule to compute hourly/daily metrics. Results stored back to Timestream.

### Native Partition-Based Multi-Tenancy

**✅ Timestream Multi-Tenancy is Superior to SiteWise**

Every Timestream table uses tenant\_id as partition key. This provides:

* Physical data separation: Each tenant's data stored in separate partitions
* Query performance: Partition pruning automatically filters data
* Security: IAM policies can restrict access at partition level
* Cost allocation: Easy to track storage and query costs per tenant
* Scalability: Partitions distribute across nodes for parallel processing

**Comparison to Other Options:**

* Options A & B (Snowflake RLS): Excellent - Database-enforced, cannot be bypassed
* Option D (Timestream partitions): Good - Native physical isolation, query-enforced
* Option C (SiteWise tags): Risky - Application-layer filtering, can be misconfigured

### Grafana for Dashboards

**Why Grafana?**

Grafana is the industry-standard visualisation platform, widely used in manufacturing and DevOps. Key advantages:

* Fully White-Labelable: Remove Grafana branding, add company logos, custom themes
* Native Timestream Plugin: Official AWS plugin for querying Timestream
* Rich Visualisation Library: Time-series charts, gauges, heatmaps, state timelines, tables
* Embeddable Dashboards: iframe embedding in custom web applications
* Alert Management: Built-in alerting with multiple notification channels
* Variable Templates: Dynamic dashboards that adapt to user selection (site, machine)
* Community Support: Massive ecosystem, extensive documentation, active forums

**Grafana Cloud (Managed):** SaaS offering from Grafana Labs. Fully managed (no servers). £10-30 per user per month depending on tier. Automatic updates. 99.9% SLA.

**Self-Hosted on ECS Fargate:** Run Grafana container on AWS ECS. Full control over deployment. One-time setup effort. Lower long-term cost for many users.

### Application State (DynamoDB)

**DynamoDB Tables:** Stores: tenant configuration, device registry, user preferences, dashboard configs, alert rules. Fast key-value lookups. Single-digit millisecond latency.

**API Gateway + Lambda:** REST APIs for: device registration, dashboard management, user settings. Enforces tenant isolation. Queries Timestream and DynamoDB.

**Cognito User Pools:** Authentication with SSO support. MFA capability. JWT tokens passed to Grafana and APIs.

**Secrets Manager:** Stores: Timestream credentials, Grafana API keys, third-party integration secrets.

## 6.4 Data Flow

Unified data flow (all data types):

1. Sensor → IoT Core via MQTT (latency: <100ms)
2. IoT Rules → Kinesis Streams (latency: <200ms)
3. Lambda processes batch → Validates, enriches (latency: <500ms)
4. Timestream Write API → Memory store (latency: <1s)
5. Grafana queries Timestream via plugin (latency: <2s)
6. Dashboard updates display (total: <5s end-to-end)

Critical alert fast-path:

1. IoT Rules detects threshold breach (e.g., CO₂ >5000 ppm)
2. Lambda fast-path triggered directly (bypasses Kinesis)
3. SNS/SES notification sent (latency: <500ms)
4. Alert logged to Timestream (latency: <2s total)

## 6.5 Strengths

**Fully AWS-Native:** No Snowflake dependency. All AWS services. Single vendor relationship. Unified billing.

**Unified Storage:** Timestream handles both continuous time-series AND discrete events. No dual-database complexity like Option C.

**Native Partition Multi-Tenancy:** Physical data isolation at partition level. Better security than SiteWise tags. Simpler than application-layer filtering.

**Industry-Standard Dashboards:** Grafana is battle-tested in manufacturing and industrial monitoring. Familiar to many operations teams. Rich visualisation library.

**Fully White-Labelable:** Remove all vendor branding. Add company logos and themes. Embed dashboards in custom portals seamlessly.

**Cost-Effective:** £3,965/month (40% cheaper than Option C). Only 15% premium over Option B for AWS-native requirement.

**Standard SQL:** Timestream SQL is mostly ANSI-compatible. Familiar to engineers. Easier to hire than Flink specialists.

## 6.6 Limitations

**15% More Expensive Than Option B:** £3,965/month vs £2,170-3,450 for Snowflake. Acceptable premium for AWS-native mandate but not cheapest option.

**Custom Asset Modelling Required:** Unlike SiteWise's built-in asset models, must build custom data schemas and calculation logic. More initial development than Option C.

**Grafana Dependency:** Whilst Grafana is popular, it adds external dependency (if using Grafana Cloud) or operational overhead (if self-hosted).

**Timestream SQL Differences:** Whilst mostly standard, some Timestream-specific syntax and functions. Learning curve for SQL engineers familiar with PostgreSQL/MySQL.

## 6.7 When to Choose

Select Option D if:

* AWS-native is mandatory (no Snowflake allowed)
* Want best-in-class time-series dashboards (Grafana)
* Need native partition-based multi-tenancy
* Comfortable with 15% cost premium vs Option B for AWS-native benefit
* Team has AWS expertise (no Snowflake learning required)
* Want unified storage for time-series + events (simpler than Option C)
* Prefer industry-standard dashboards over custom development

**✅ RECOMMENDED** if AWS-native is a requirement. Option D provides the best AWS-native solution without SiteWise's multi-tenancy limitations or custom dashboard burden. It is the second-best overall option after Option B (Snowflake).