18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.

SpaceOps-2025, ID # 180

Real-Time Anomaly Detection in Satellite Constellation Telemetry using AWS

Ed Meletyan[a]*, Eric Parsell[b]

[a] *Amazon Web Services (AWS), United States*, *emelet@amazon.com*
[b] *Amazon Web Services (AWS), United States*, *eparsell@amazon.com*
* Corresponding Author

Abstract

The rapid growth of commercial satellite constellations has created unprecedented challenges in processing the massive volumes of telemetry data generated by thousands of spacecraft. This paper presents a cloud-native architecture for real-time telemetry processing and anomaly detection that scales efficiently with constellation size. The solution combines Amazon Managed Streaming for Apache Kafka (MSK) for data ingestion, AWS Lambda for stream processing, and Amazon SageMaker for machine learning inference using a Random Cut Forest model. Testing with constellation sizes ranging from 5 to 5000 satellites demonstrates consistent sub-100ms inference latencies (mean 74ms, standard deviation 44ms) and 100% processing reliability. The system processes telemetry at 10 Hz per satellite through a three-partition Kafka configuration, performing inference on up to 2.0×10^5 unique telemetry points per second. The architecture maintains stable performance from small-scale testing through full constellation deployment, with linear scaling in both processing capability and cost. The solution successfully addresses three critical challenges in modern satellite operations: scalable ingestion of high-volume telemetry streams, real-time processing for anomaly detection, and efficient data persistence. By leveraging serverless computing and managed services, the system eliminates the need for infrastructure provisioning while providing consistent performance across varying constellation sizes. These results demonstrate the viability of cloud-native architectures for spacecraft operations and provide a blueprint for implementing scalable telemetry processing systems across the space industry.

Keywords: satellite operations, cloud computing, anomaly detection, serverless architecture, telemetry processing

Nomenclature

φ	Phase angle of satellite [radians]
n	Number of satellites in constellation
M_a	Anomaly magnitude multiplier [dimensionless]
x_n	Nominal telemetry value [varies by parameter]
x_a	Anomalous telemetry value [varies by parameter]

Acronyms/Abbreviations

Amazon Managed Streaming for Apache Kafka (MSK)
Amazon Simple Storage Service (S3)
Amazon Web Services (AWS)
Central Processing Unit (CPU)
Identity and Access Management (IAM)
Low Earth Orbit (LEO)
On-board Compute (OBC)
Random Cut Forest (RCF)

1. Introduction

The proliferation of commercial satellite constellations has fundamentally transformed space-based operations, generating unprecedented volumes of telemetry data that challenge traditional ground system architectures. Some constellations can generate between 80 terabytes of data per day [1], a combination of mission data and telemetry. This exponential growth in data volume necessitates a paradigm shift in how operators process, analyze, and store spacecraft telemetry.

Traditional on-premises data centers, typically designed for predictable workloads, face significant limitations when managing the variable nature of satellite telemetry streams. These systems often require substantial over-provisioning to handle peak loads, leading to inefficient resource utilization during normal operations. Moreover, the real-time nature of spacecraft operations demands immediate processing and analysis of telemetry data to enable rapid

response to anomalous conditions, a requirement that becomes increasingly difficult to meet as constellation sizes grow.

Recent advances in cloud computing and streaming architectures offer new approaches to these challenges. Stream processing technologies, particularly Apache Kafka, have demonstrated capability in handling high-throughput, real-time data streams in various industries. However, implementing and maintaining these technologies in a space operations context presents unique challenges, including the need for guaranteed message delivery, strict ordering requirements, and real-time anomaly detection across thousands of spacecraft simultaneously.

This research addresses three critical challenges in modern satellite operations: the scalable ingestion of high-volume telemetry streams, real-time processing for anomaly detection, and efficient data persistence for long-term analysis. The solution implements a cloud-native architecture that leverages managed streaming services, serverless computing, and machine learning to process telemetry data at constellation scale. By utilizing MSK for data ingestion, AWS Lambda for stream processing, and Amazon SageMaker for real-time inference, the system achieves horizontal scalability while maintaining consistent processing latency across varying constellation sizes.

The significance of this work extends beyond immediate operational benefits, presenting a framework for processing spacecraft telemetry that scales linearly with constellation size without requiring changes to the underlying architecture. This approach enables operators to focus on mission operations rather than infrastructure management, while maintaining the flexibility to accommodate future growth in both data volume and processing requirements.

The following sections detail the system architecture, theoretical foundations of the stream processing implementation, and quantitative analysis of the solution's performance across multiple scale points from 5 to 5000 satellites. The results demonstrate the viability of cloud-native architectures for spacecraft operations and provide a blueprint for implementing similar systems across the space industry.

2. Materials and Methods

2.1 System Architecture

The architecture implements a real-time telemetry processing pipeline optimized for satellite constellation operations. While the solution includes data archival capabilities through Amazon Kinesis Firehose to S3, this paper focuses on the real-time anomaly detection pathway. The architecture's modular design enables independent scaling of each component based on constellation size and telemetry volume, while maintaining consistent processing latency across the system. Figure 1 outlines the AWS services used in this architecture.

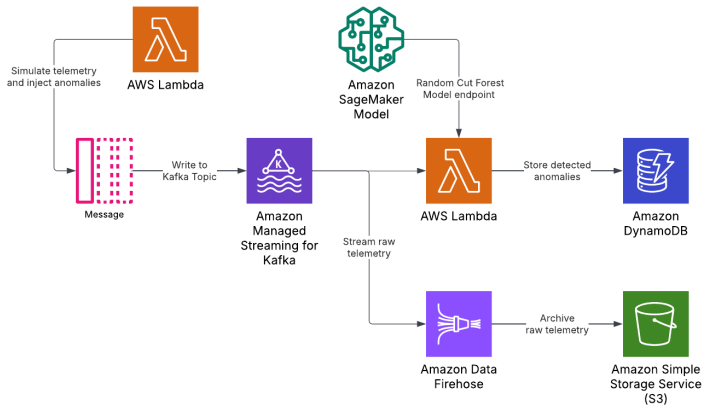

Figure 1. Cloud-native architecture for performing anomaly detection on simulated satellite telemetry.

The solution consists of five primary components. An AWS Lambda function simulates spacecraft telemetry and injects anomalies at configurable rates, streaming data to MSK at 10 Hz per satellite. Apache Kafka is a popular choice for real-time streaming [2] and MSK provides the streaming backbone through a three-partition Kafka cluster, offering write throughput of 200 MiB/second and read throughput of 400 MiB/second. A second AWS Lambda function processes the telemetry stream and invokes a SageMaker endpoint hosting a random cut forest model for anomaly detection. Detected anomalies are persisted to DynamoDB for operator notification and analysis.

2.2 Telemetry Production

The producer component implements secure access to the Kafka cluster through AWS IAM authentication. It manages topic creation and partition configuration automatically, enabling parallel processing while maintaining

message ordering within each partition. The system incorporates automatic retry logic with a 500ms backoff period and 20-second request timeout to ensure reliable message delivery.

The telemetry generation system operates at 10 Hz per spacecraft, with timing controlled through environment variables that specify message count, topic configuration, and execution duration. This configurable approach enables testing across various operational scenarios and constellation sizes.

2.3 Anomaly Detection Implementation

The anomaly detection system combines controlled anomaly injection with real-time inference. The producer allows configuration of both anomaly frequency (0% – 100% of produced telemetry) and magnitude, enabling systematic testing of detection capabilities. The consumer component processes incoming telemetry streams and interfaces with a SageMaker endpoint hosting a random cut forest model for anomaly detection.

2.4 Performance Monitoring

The system implements comprehensive latency tracking throughout the processing pipeline. For each telemetry batch, the system measures and records duration of machine learning model execution and pipeline execution time

Each telemetry record generates a response containing the input value, calculated anomaly score, and associated timing metrics. This instrumentation enables detailed performance analysis across varying constellation sizes and telemetry volumes.

2.5 Data Persistence

The system implements dual-path data handling. The primary path focuses on real-time anomaly detection, with results stored in DynamoDB for immediate operator access. A secondary path, implemented through Kinesis Firehose, archives raw telemetry to S3 for historical analysis and model retraining. This approach balances real-time operational needs with long-term data retention requirements.

3. Theory and Calculation

3.1 Telemetry Generation Model

The solution implements a physics-based telemetry generation model that simulates spacecraft health and location metrics. The model accounts for orbital dynamics and power system characteristics typical of LEO satellites. For a constellation of n satellites, the system generates synchronized telemetry streams with phase angles φ_i distributed uniformly around each orbital plane:

$$\varphi_i = \frac{2\pi i}{n}, \qquad i \in \{0, \ldots, n-1\} \tag{1}$$

3.2 Anomaly Injection Methodology

The system injects anomalies into the telemetry stream using a probabilistic approach. For each telemetry point, the probability of nominal behavior is defined by the anomaly rate parameter. When an anomaly is triggered, the system applies a magnitude multiplier to the baseline telemetry values:

$$x_a = x_n * (1 + M_a/100) \tag{2}$$

This approach creates detectable deviations while maintaining physically realistic values.

3.3 Random Cut Forest Model

The solution employs a Random Cut Forest (RCF) algorithm for real-time anomaly detection in spacecraft telemetry streams. RCF operates as an unsupervised learning algorithm particularly suited for streaming applications. The primary focus of this paper is to explore the speed and scalability of infrastructure that leverages machine leaning models. Model training and selection are outside the scope of this work but there are numerous resources available that outline how to train and deploy them [3][4].

3.4 Latency Analysis

The system calculates three critical timing metrics for each telemetry batch including time to consume a message, time to perform inference, and total latency from production through inference. These metrics enable quantitative analysis of system performance across different constellation sizes and telemetry volumes.

4. Results

4.1 System Performance Analysis

The solution's performance was evaluated across multiple constellation sizes, ranging from 5 to 5000 satellites, with each satellite generating telemetry at 10 Hz. Figure 2 presents the inference latency distribution across these scale points.

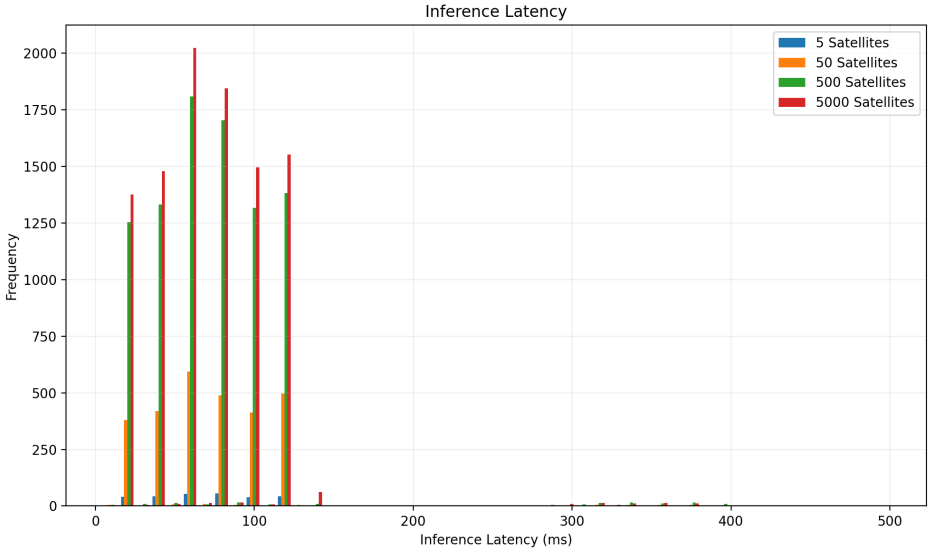

Figure 2. Inference latency stability with increasing number of telemetry streams.

A key finding is the consistent inference latency regardless of constellation size. The SageMaker endpoint maintained sub-100ms average response times across all test cases, with a mean latency of 74 ms and standard deviation of 44 ms. There is a batch of latency results that are greater than 300 ms. These values occur during “cold starts” of the infrastructure while the service is provisioned. The stability of performance demonstrates the solution's ability to scale horizontally without degrading latency.

4.2 Streaming Infrastructure Performance

The MSK implementation demonstrated robust performance characteristics during testing. Figure 3 illustrates the throughput metrics recorded during peak operation.

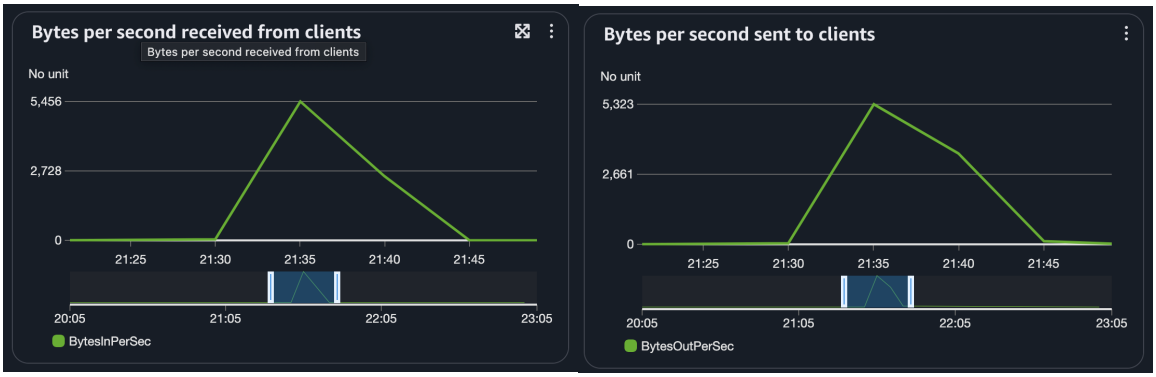

Figure 3. MSK dashboard showing bytes per second received and sent across clients over the test period 21:30 to 21:45.

The three-partition Kafka configuration efficiently handled the increasing data volume, maintaining consistent throughput as the producer AWS Lambda functions scaled from simulating 5 to 5000 spacecraft. The system achieved stable performance and was able to run inference on 2.0×10^5 messages per second where each message was an average of 90 bytes and contained 4 unique telemetry points (altitude, OBC memory usage, OBC CPU usage, and battery level). This throughput is 9% of MSK Serverless service limits of 200 MiB/second write throughput and 400 MiB/second read throughput which ensures that additional telemetry points would not impact system availability.

4.3 System Scalability

The producer AWS Lambda functions demonstrated linear scaling capabilities, as shown in Figure 4. Each AWS Lambda invocation produces telemetry from 5 spacecraft.

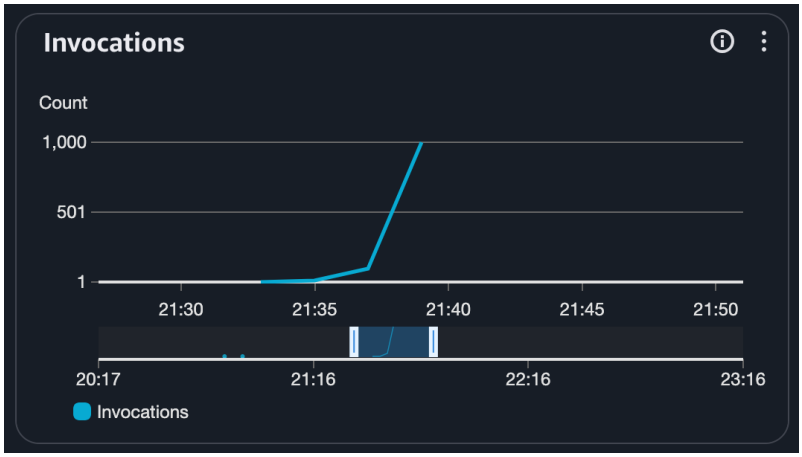

Figure 4. AWS Lambda invocations scaling from 1 to 1000 producers, each simulating 5 spacecraft.

The system maintained consistent performance as the number of producer functions increased, with no observable degradation in message delivery or processing time. Each producer function successfully maintained the required 10 Hz telemetry generation rate for its assigned spacecraft.

4.4 Reliability Analysis

Throughout the test period, the system demonstrated robust processing reliability. Figure 5 shows success metrics output from the AWS Lambda console.

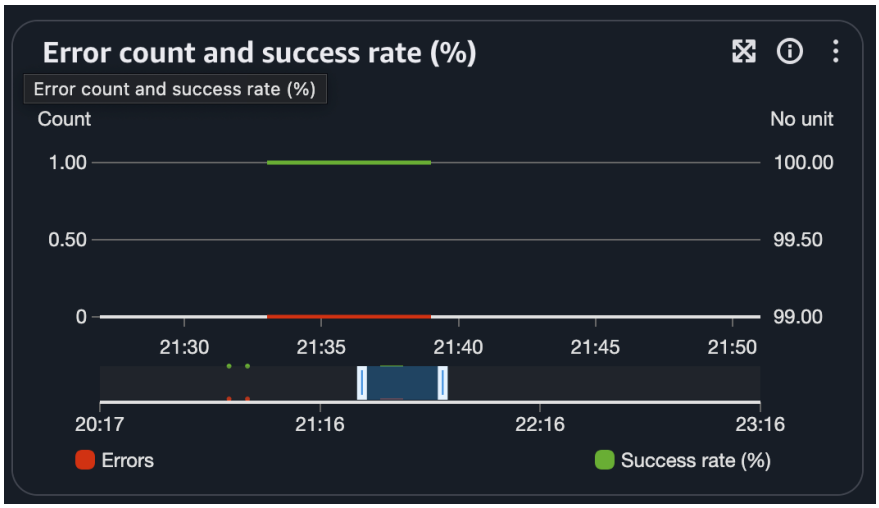

Figure 5. Recorded success rates for the processing and inferencing pipeline

The consumer AWS Lambda functions recorded zero processing errors across all constellation sizes, achieving 100% successful message processing. This reliability extends across the entire processing pipeline, from telemetry generation through anomaly detection and persistence.

5. Discussion

The experimental results reveal significant insights about cloud-native architectures for spacecraft operations. The system's performance characteristics demonstrate both the advantages and potential limitations of this approach for constellation management.

The consistent sub-100ms inference latency across all constellation sizes represents a significant advancement over traditional ground system architectures. While conventional systems typically show degraded performance as telemetry streams increase, this solution maintained stable processing times even with 5000 telemetry streams. This consistency stems from the three-partition Kafka configuration, which provided sufficient parallelism without over-complicating the stream processing topology. SageMaker endpoint scaling handled increased inference loads without requiring manual intervention, while AWS Lambda's automatic scaling effectively distributed processing across available resources.

The zero-error rate in consumer functions validates the effectiveness of cloud service management layer. The linear scaling of producer functions demonstrates the architecture's elasticity, but also highlights potential cost considerations. The system efficiently handles increased load while the pay-per-use model ensures systems optimize operational costs by not over- or under-provisioning infrastructure.

The RCF model's consistent performance across varying constellation sizes validates its selection for streaming telemetry analysis. The model's ability to process multi-dimensional telemetry without increased latency enables comprehensive health monitoring, while its unsupervised learning capability eliminated the need for extensive training data. The compact memory footprint allowed efficient deployment across multiple AWS Lambda instances. With inferences occurring at a fraction of the 100 ms production rate, this architecture adheres to a real-time processing paradigm.

Looking forward, the results suggest several areas for potential enhancement. The implementation of a machine learning model tuned to the telemetry of specific spacecraft would improve anomaly detection performance [5][6]. Integration of automated response mechanisms for detected anomalies would enhance operational efficiency by notifying operators or performing artificial intelligence-powered autonomous maintenance. Future work should also focus on improving the fidelity of virtualized telemetry producers. Satellite digital twins can integrate with this architecture to validate the system using operational data.

These findings demonstrate that while cloud-native architectures offer compelling advantages for constellation operations, successful implementation requires careful consideration of both technical and operational factors. The solution's performance characteristics suggest it could serve as a template for future ground system architectures, particularly as constellation sizes continue to grow.

6. Conclusions

The research demonstrates the viability of cloud-native architectures for processing satellite telemetry at constellation scale. The implementation successfully processed telemetry from 5 to 5000 satellites while maintaining consistent sub-100ms inference latencies and achieving zero processing errors. The solution's ability to scale horizontally without performance degradation validates the serverless approach for space operations.

The combination of MSK, AWS Lambda, and SageMaker services created a robust processing pipeline capable of handling high-frequency telemetry streams. The three-partition Kafka configuration proved sufficient for managing telemetry from thousands of satellites, while the RCF model effectively detected anomalies in multi-dimensional telemetry data without requiring pre-labeled training data. This architecture maintained consistent performance from small-scale testing through full constellation deployment, with linear scaling in both processing capability and cost.

Performance metrics demonstrate significant advantages over traditional ground system architectures. The solution processed telemetry at 10 Hz per satellite with consistent latency, independent of constellation size. The streaming architecture efficiently handled increasing data volumes, maintaining stable throughput well within the system's theoretical limits while performing inference on 2.0×10^5 unique telemetry values. This performance stability, combined with pay-per-use pricing, offers satellite operators a cost-effective alternative to traditional fixed infrastructure.

Future work should focus on several key areas. These areas include fine-tuning RCF models with operational telemetry, adding automated pipelines for handling detected anomalies (autonomous maintenance and operator notification), and connecting high-fidelity satellite emulators to generate realistic telemetry.

The significance of this research extends beyond the immediate technical implementation. As the satellite industry continues its rapid growth, the demonstrated ability to process constellation-scale telemetry using cloud-native architectures provides a blueprint for future ground system design. This approach enables operators to focus on mission operations rather than infrastructure management, while maintaining the flexibility to accommodate future growth in both constellation size and operational complexity.

These results suggest that cloud-native architectures represent not just a viable alternative to traditional ground systems, but potentially a new paradigm for spacecraft operations as the industry moves toward increasingly large constellations. The combination of scalability, reliability, and cost-effectiveness demonstrated by this solution provides a foundation for the next generation of space operations infrastructure.

References

[1] J. Carr, “Sending Data From Space to Amazon S3 in Less Than a Minute,” Maxar Blog, Nov. 27, 2018. https://blog.maxar.com/earth-intelligence/2018/sending-data-from-space-to-amazon-s3-in-less-than-a-minute

[2] M. R. Sayem and M. Oguike, “Build an end-to-end Serverless Streaming Pipeline with Apache Kafka on Amazon MSK Using Python | Amazon Web Services,” AWS Big Data Blog, Mar. 21, 2024. https://aws.amazon.com/blogs/big-data/build-an-end-to-end-serverless-streaming-pipeline-with-apache-kafka-on-amazon-msk-using-python/ (accessed Apr. 06, 2025).

[3] “Random Cut Forest (RCF) Algorithm - Amazon SageMaker,” docs.aws.amazon.com. https://docs.aws.amazon.com/sagemaker/latest/dg/randomcutforest.html

[4] C. Swierczewski and L. Jiang, “Using Random Cut Forests for real-time Anomaly Detection in Amazon OpenSearch Service,” AWS Big Data Blog, Jun. 05, 2020. https://aws.amazon.com/blogs/big-data/using-random-cut-forests-for-real-time-anomaly-detection-in-amazon-opensearch-service/ (accessed Apr. 06, 2025).

[5] S. Jiang, Y. Jiang, Y. Wang, X. Zhang, and Z. Zhang, “Anomaly Detection in Spacecraft Telemetry Data Based on Transformer-LSTM,” IEEE, pp. 271–276, Nov. 2023, doi: https://doi.org/10.1109/icn60549.2023.10426290.

[6] J. He, Z. Cheng, and B. Guo, “Anomaly Detection in Satellite Telemetry Data Using a Sparse Feature-Based Method,” Sensors (Basel, Switzerland), vol. 22, no. 17, p. 6358, Aug. 2022, doi: https://doi.org/10.3390/s22176358.