WHITE PAPER

EVENT DRIVEN MICROSERVICES WITH APACHE KAFKA AND OTHER STREAMING FRAMEWORKS – A FINANCIAL SERVICES PERSPECTIVE

Abstract

Event-driven microservices leveraging data-streaming backbone fulfills the real promise of containerized microservices towards organizational agility & strategic business value. Advances in data-streaming frameworks open new avenues for microservices adoption and render event-driven microservices as an appealing design choice for businesses.

Financial services need to embrace this revolution for improving their agility, customer experience, risk management & scalability. This Point of View discusses the case for event-driven microservices with focus on financial services, relevance of data streaming, architecture blueprint, considerations, a few significant patterns in implementation of event driven microservices and microservices leveraging data streaming solutions.

Infosys®
Navigate your next

Events First Strategy – Revolutionizing Financial Services

Why FS organizations need an event-first strategy?

Events are the first-class citizens of every Financial Services (FS) organization. Every transaction in FS domain consists of multiple events which represent facts. Price fluctuations in exchange traded funds, stock movements, or simple transactions like account opening carry an enormous significance if these events are captured when they are generated and analyzed in real time. For example, two credit card transactions within an interval of ten minutes, from the same credit card from two different merchant outlets situated a few miles apart can be potentially fraudulent. Real-time analysis of stock movements can provide proactive insights to traders to safeguard customer investments.

As new services are introduced or new regulations are enforced, organizations need to do a tight-rope walk between the two ends – 'a digital-savvy experience' on one side and 'increasingly stringent regulatory reporting' on the other. An urge to move out of costly mainframes to become cloud native & super-elastic further aggravates the need for a streamlined & reactive IT architecture. Modelling an organization around the events allows IT architectures to evolve without friction across business groups. This is a critical capability for any financial enterprise as business groups often work in silos. Advances in streaming data analytics amplify this benefit by supporting real-time decision making within the framework of regulatory compliance.

Advantages of data streaming for FS organizations

Every financial organization now maintains its Data Lake & Data Warehouse. But by the time the data has been stored there, it loses its time value. Data streaming enables real time analytics as the events are generated. For example, ATM cash withdrawal events can help a bank optimize its cash recharge operations. Data streaming architecture has a nimble footprint and it scales in real time allowing faster returns on investments.

For any bank, offering personalized services based on customer profile and customer behavior or location is always a challenge. A data streaming platform can capture trends of customer behavior, use machine learning to analyze patterns & helps deliver a personalized digital experience. Customer behavior across digital channels, websites, ATMs, or even legacy systems can be aggregated as event streams & analyzed to produce an accurate view of customer profile, segment, risk etc.

Event streaming offers valuable insights for regulatory reporting since complete lineage is possible between decisions taken at a given point of time and the underlying data that existed at that time. It is possible because events are persisted in fault-tolerant & scalable data pipelines in the order in which they occur. Such events can even be replayed to simulate the system state. For example, a machine learning algorithm can scan capital market movements across data pipelines and correlate them against a bank's profit & loss position to devise an optimal data analytics model.

Real time insights into events produced by financial or non-financial transactions empower FS organizations to design a future ready risk management platform. Huge volumes of real time data can be leveraged to tune machine learning models for intelligent threat monitoring. Customer clicks on payment websites, typical locations of cash withdrawal ATMs, cash withdrawal patterns etc. can be analyzed instantaneously to challenge a fraudulent transaction.

Example scenarios for FS organizations leveraging event streaming

Organizations cannot survive unless they remain ahead of their customers – by delivering personalized content at the most appropriate customer interaction touchpoints. It necessitates building real time insights into customer preferences and intelligent analytics to deliver outcome focused customer experience. An event driven platform as shown below can gather multiple facets of customer engagement at runtime, enrich them with segment and product-specific information and feed into predictive analytics model to drive customer engagement.

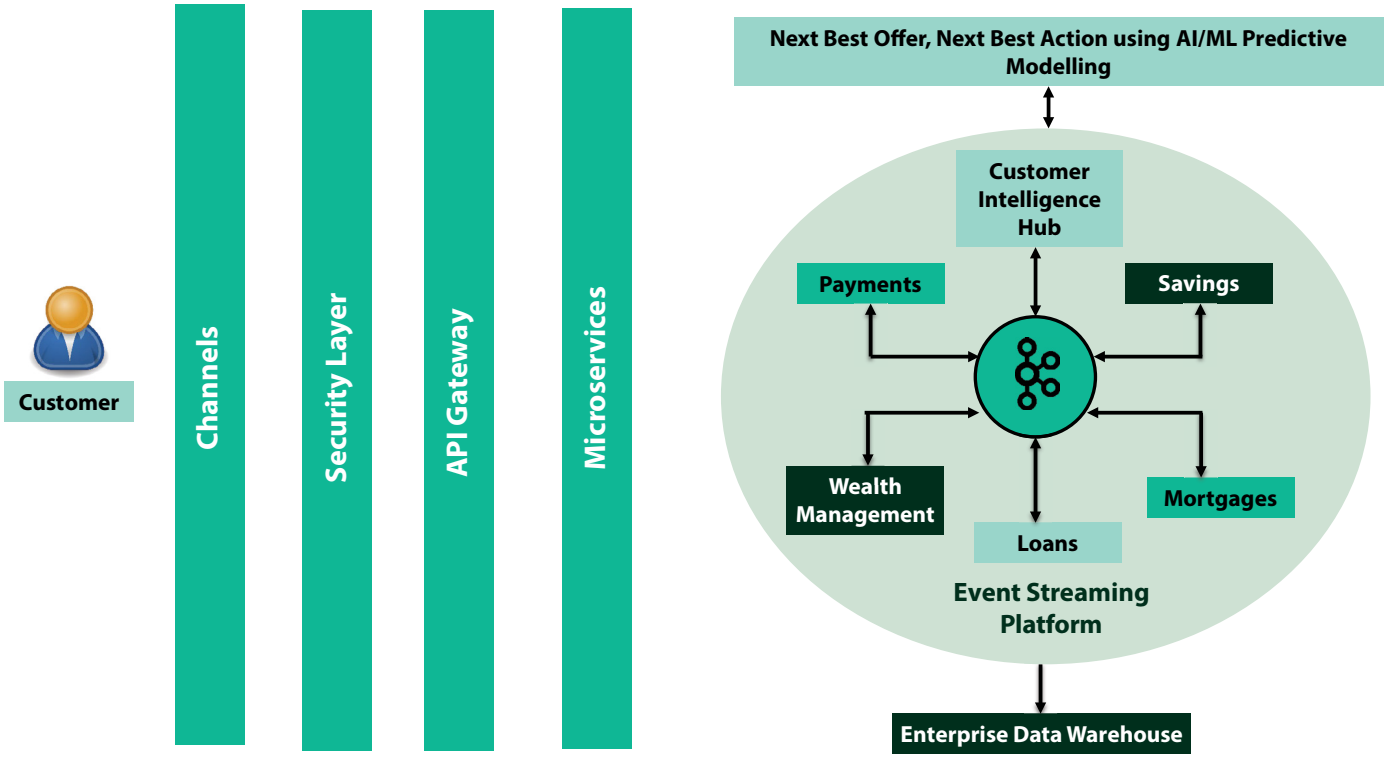


Figure 1: Digital Engagement Hub with Customer 360° View

With ever increasing financial vigilance all over the world, financial institutions are always on their toes to keep their system of records up to date to meet regulatory submission timelines and stringent data integrity checks across multiple regulatory reports. Apart from the need to have a robust data foundation layer, financial institutions need an agility for onboarding new regulatory reports. Onboarding of regulatory reports needs co-ordination across various groups which are often disconnected. The group that onboards regulatory reports is often different from the group that creates rules to generate financial figures for those reports, report generation & submission are again handled by separate groups. This restricts the ability to respond efficiently to changes in reporting standards.

Event streaming based architecture acts as a central nervous system that glues together heterogeneous data sources, ensures highly integrated reporting data, and automates entire regulatory report generation & submission process as illustrated in the below diagram.

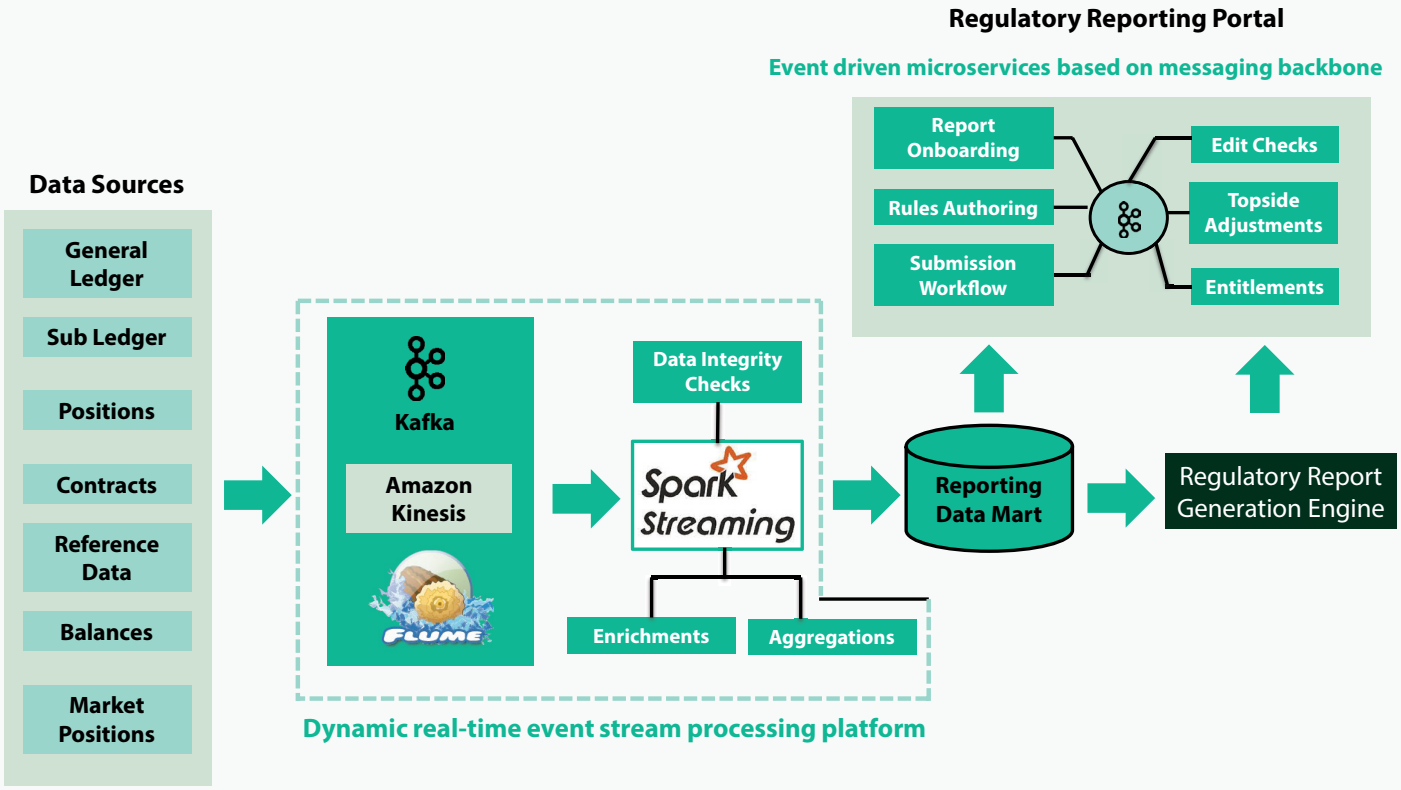


Figure 2: Regulatory Reporting Platform leveraging Event Stream Processing

The list of applications in financial services, where one can leverage event streaming is ever evolving. Let us now take a quick look at technical aspects of event streaming based microservices.

Event Driven Microservices

What is event-driven microservices with data streaming?

Recognizing the business value of real-time streaming data processing, technology has evolved from Big Data at rest (HDFS – Hadoop Distributed File System) to Distributed Streaming. Open-source technologies such as Apache Kafka, Apache Spark, Apache Samza and proprietary solutions such as Amazon Kinesis, Google Cloud Data Flow coupled with microservices enable event-driven microservices that provide super-elastic, universal & persistent connectivity to generate, process & persist streaming data. This combination of event-driven microservices with simple & robust streaming backbone fulfills the real promise of microservices architecture by greatly enhancing the organizational agility to build, deploy & maintain data-streaming pipelines to support cloud-native, highly available & performant business solutions.

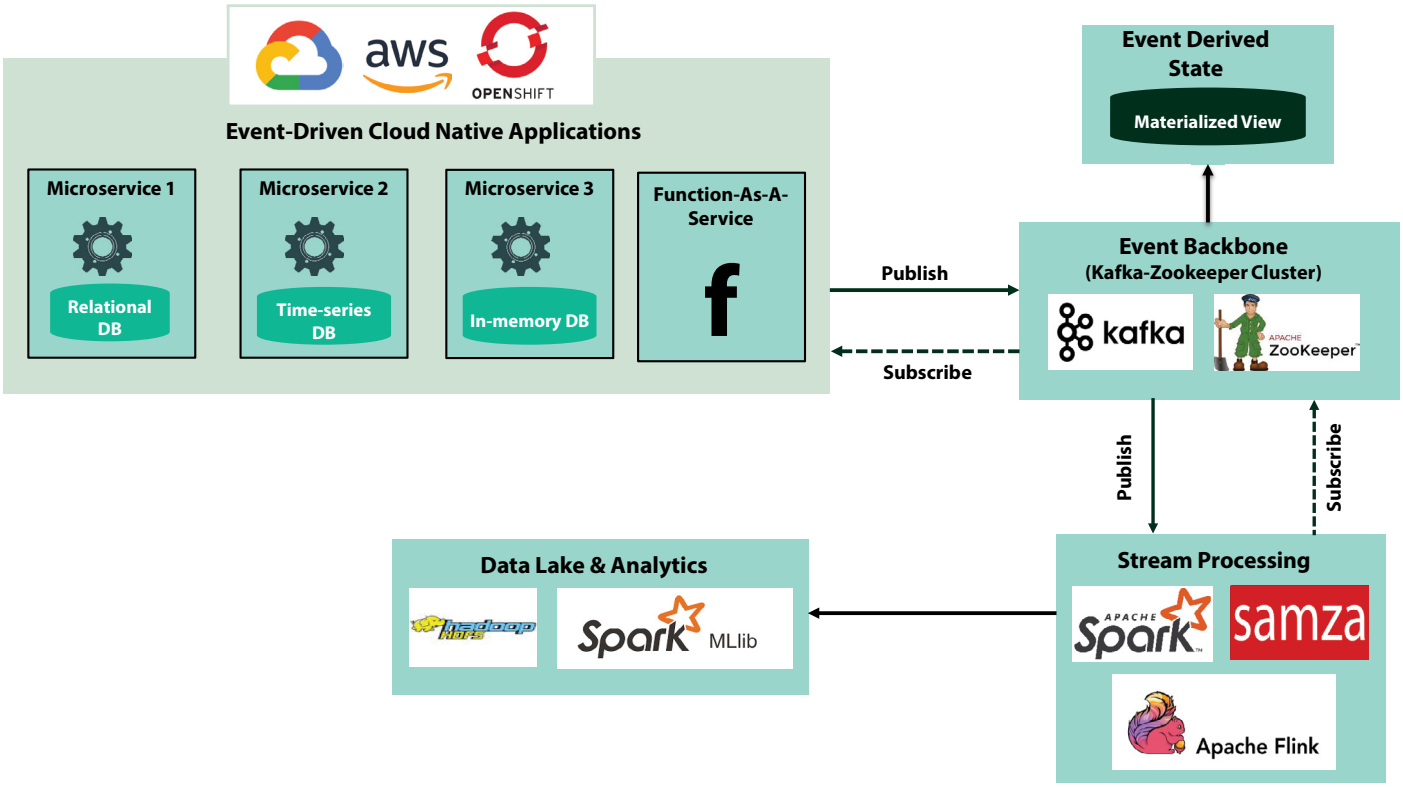


Figure 3: Overview of Event Driven Microservices leveraging Data Streaming

Advantages of event-driven architectures

- De-coupling between event producers & event consumers
- Resiliency
- Auditability of System State by design of the system itself
- Online Analytics by leveraging data streaming pipelines & machine learning
- Accelerating Machine Learning Models Deployment

High Level Architecture

Reference architecture for an Event-driven Microservices architecture-based system leveraging data streaming backbone is given below.

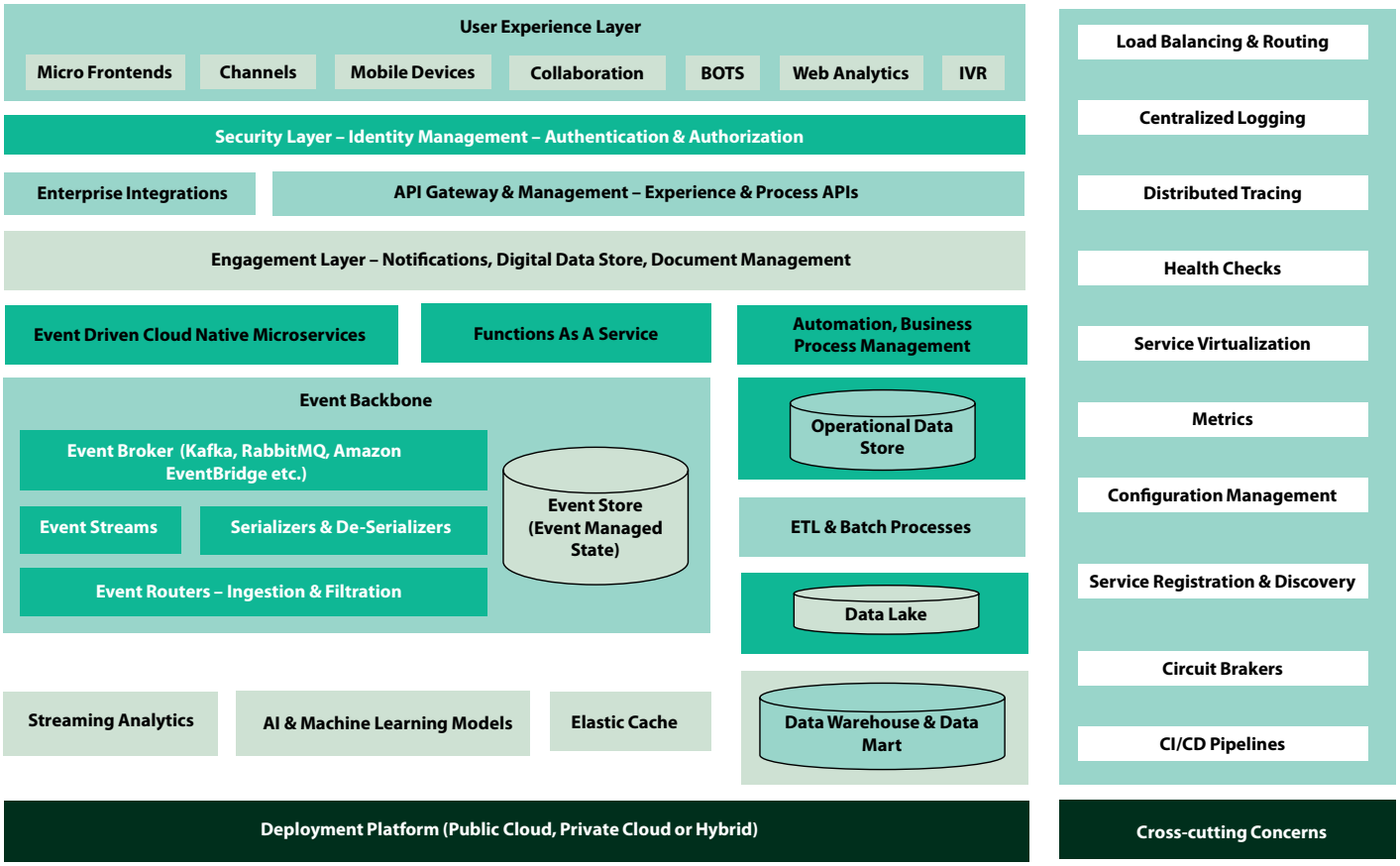


Figure 4: Reference Architecture - Event Driven Microservices leveraging Data Streaming

Technology Choice for Event Backbone

Horizontal scalability of event backbone is the first critical aspect for event driven microservices which work with streaming data. Peaks in event generation should be handled by seamlessly adding more capacity & processing power without system downtime.

Events generated from edge devices across geographies have different time-zones. Unreliable networks further complicate the issues since an event generated earlier may be available on event backbone much later due to network issues. Hence strong ordering guarantee of event backbone is the second critical requirement for event driven microservices.

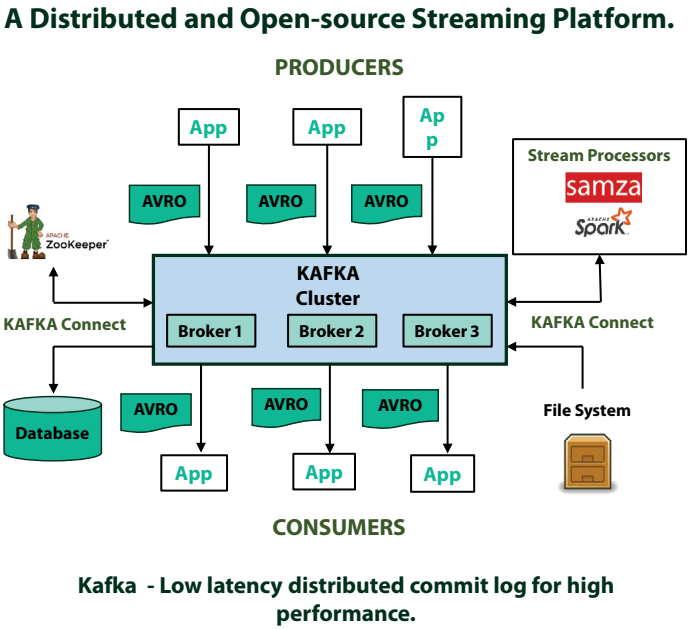
There exists a trade-off between data replication & data integrity. While clustering is essential for data replication, data read from one cluster may have already become stale across another cluster of event backbone. Hence consistency & durability is the third critical requirement that must be satisfied by an event backbone for event driven microservices. Availability on PaaS (Platform as a Service) are other essential requirements of event backbone.

Some of the most prominent technology choices for event backbone are as below:

- Apache Kafka
- Amazon EventBridge , Amazon SNS, Amazon Kinesis
- Google Cloud: Events for Cloud Run for Anthos, Dataflow for Complex Event Processing (transformation, enrichment on streaming data), Eventarc
- Red Hat AMQ Streams
- Kafka on Kubernetes with Strimzi, Akka Streams
- IBM Event Streams on the IBM Cloud Pak

Apache Kafka – Distributed Streaming Platform

Apache Kafka is the ubiquitous technology for event driven microservices.



PUBLISH/SUBSCRIBE

Publishes (distributes) and subscribes (consumes) stream of records.

PROCESS

Super scalable stream processing in real-time.

STORE

Persists streams of data immutably in a distributed & replicated cluster offering fault tolerance.

- **Topic is the principal component that Kafka provides to store a stream of records.**
- **Kafka partition is like a safe vault – it is immutable. It provides strong ordering guarantee. Messages are continuously appended to the end of the partition.**
- **Kafka topic consists of one or more partitions. Each partition must fit into the file system of the host machine. Kafka topic is super scalable as it can span across multiple such partitions.**
- **Kafka supports geo-replication - messages are replicated across multiple datacenters.**
- **Kafka can process a few terabyte of persistent data as well as it can process a few kilobyte of data.**

Figure 5: Apache Kafka – Distributed and Open Source Streaming Platform

Event driven microservices using Apache Kafka & Apache Samza

Apache Samza is an open-source distributed stream processing framework which is targeted towards real-time streaming analytics with near-zero latencies. The beauty of Samza lies in the fact that Samza tasks can be written as plain Java applications which support integration with **Apache Kafka, AWS Kinesis, Azure Event Hubs, Elasticsearch & Apache Hadoop.** Samza supports stateless as well as stateful processing. Support for stateful processing ensures that Samza tasks can maintain integrity of their in-process state and resume stream processing from the point where a failure occurred. Samza supports both in-order & out-of-order event processing. Samza supports fully asynchronous API for high throughput remote I/O & provides "at-least once message delivery guarantee".

Kafka Streams is a Java client library for streams processing. Plain Java applications can be enriched with stream processing features by using Kafka Streams API. Kafka Streams library makes it a reality to design mission critical streaming applications in plain Java but with all advantages of Kafka. Kafka Streams recognizes table & stream duality to allow developers model streams as tables.

Strimzi is an open source project for executing Apache Kafka on Red Hat OpenShift platform & Kubernetes.

The combination of Apache Samza, Kafka Streams & Strimzi can be used to design powerful event driven architecture based upon microservices & real time streaming analytics as depicted in below diagram.

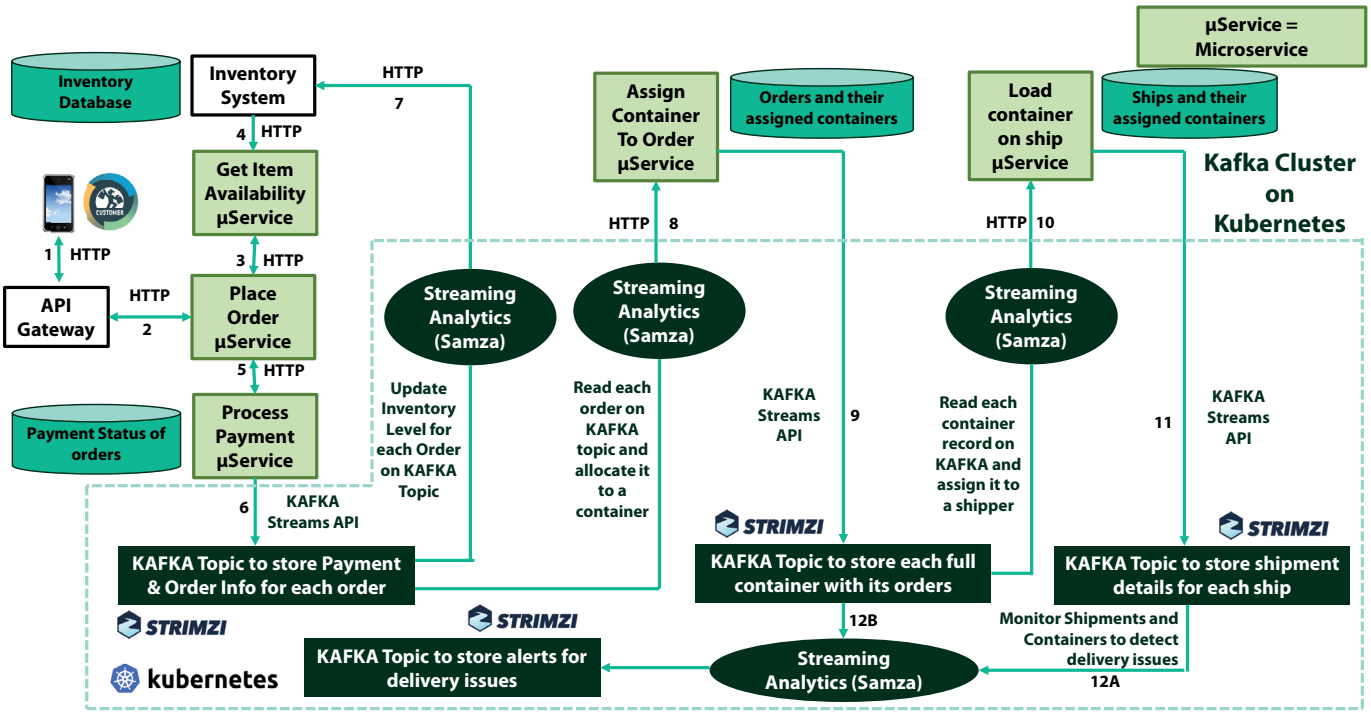


Figure 6: Real Time Streaming Analytics – Customer Order Management using Microservices, Apache Samza & Apache Kafka

Event driven microservices using Apache Flink

Pattern Recognition is one of the most prominent use-cases in machine learning. Training of machine learning models is a data hungry affair and that is how Big Data relates to machine learning. Pattern recognition requirement also means that machine learning programs should analyze streaming data in real-time. Hence there is a very close relationship between Machine Learning, Big Data & Streaming Analytics. Only those organizations which keep customer behaviors before their products can survive and that requires lot of real time data analytics and machine learning or complex event processing. That is where the value of **Apache Flink** lies. Flink implements both Lambda & Kappa architectures for supporting batch analytics & real-time analytics respectively.

First generation for Big Data processing was based on Map Reduce & Hadoop. Apache Spark was the second-generation technology for Big Data. The third generation is led by Apache Flink which is an open source streaming analytics framework & engine. It supports stateful stream processing at scale over bounded & unbounded streams. State is persisted in RocksDB, an embedded key-value store. Flink, unlike Samza guarantees exactly once message-delivery semantics if event source implements snapshots. Apart from YARN & Mesos support, Flink can be deployed natively on Kubernetes to support cloud native architecture.

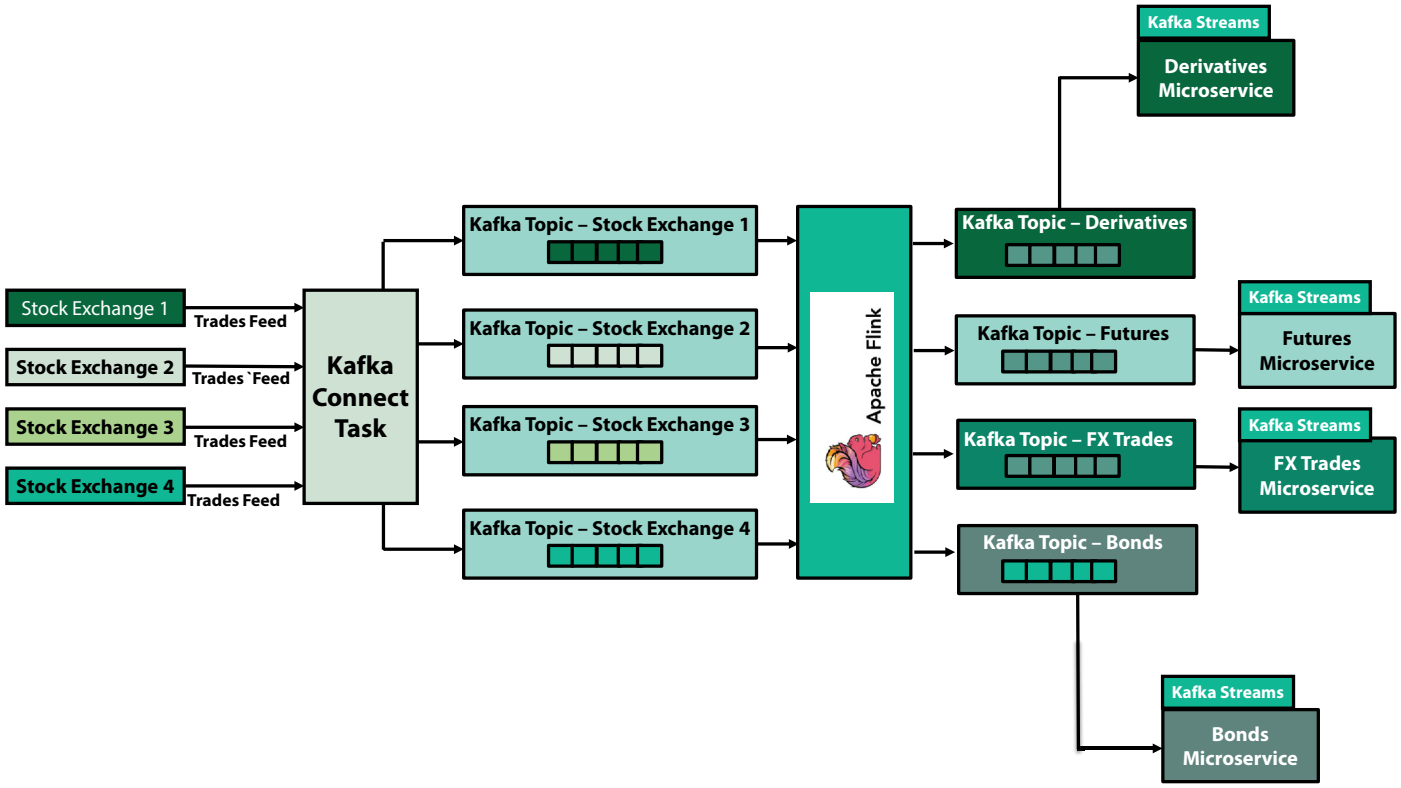


Figure 7: Real Time Stream Processing – Microservices, Kafka & Flink

Event Driven Microservices with Streaming on Amazon Web Services

Amazon Kinesis Data Firehose is the streaming component of Amazon AWS toolset. It is a fully managed service with auto-scaling support. It supports capturing of streaming data, transformations & delivering it to Amazon S3, Amazon Redshift or Amazon Elasticsearch.

As depicted in the diagram below, we have 4 microservices consisting of API gateway, AWS Lambda & AWS DynamoDB. "Receive Insurance Application Microservice" undergoes a state change when it receives a new application for insurance. It publishes an event to **Amazon Kinesis Data Streams.** Kinesis Data Streams application that runs into each of the four AWS Lambda handles message filtration, processing & forwarding. **Amazon S3** Bucket acts as the persistence store for all events.

Amazon CloudWatch dashboard can monitor aspects such as data buffering or ingestion, based on the metrics set for Kinesis Data Firehose.

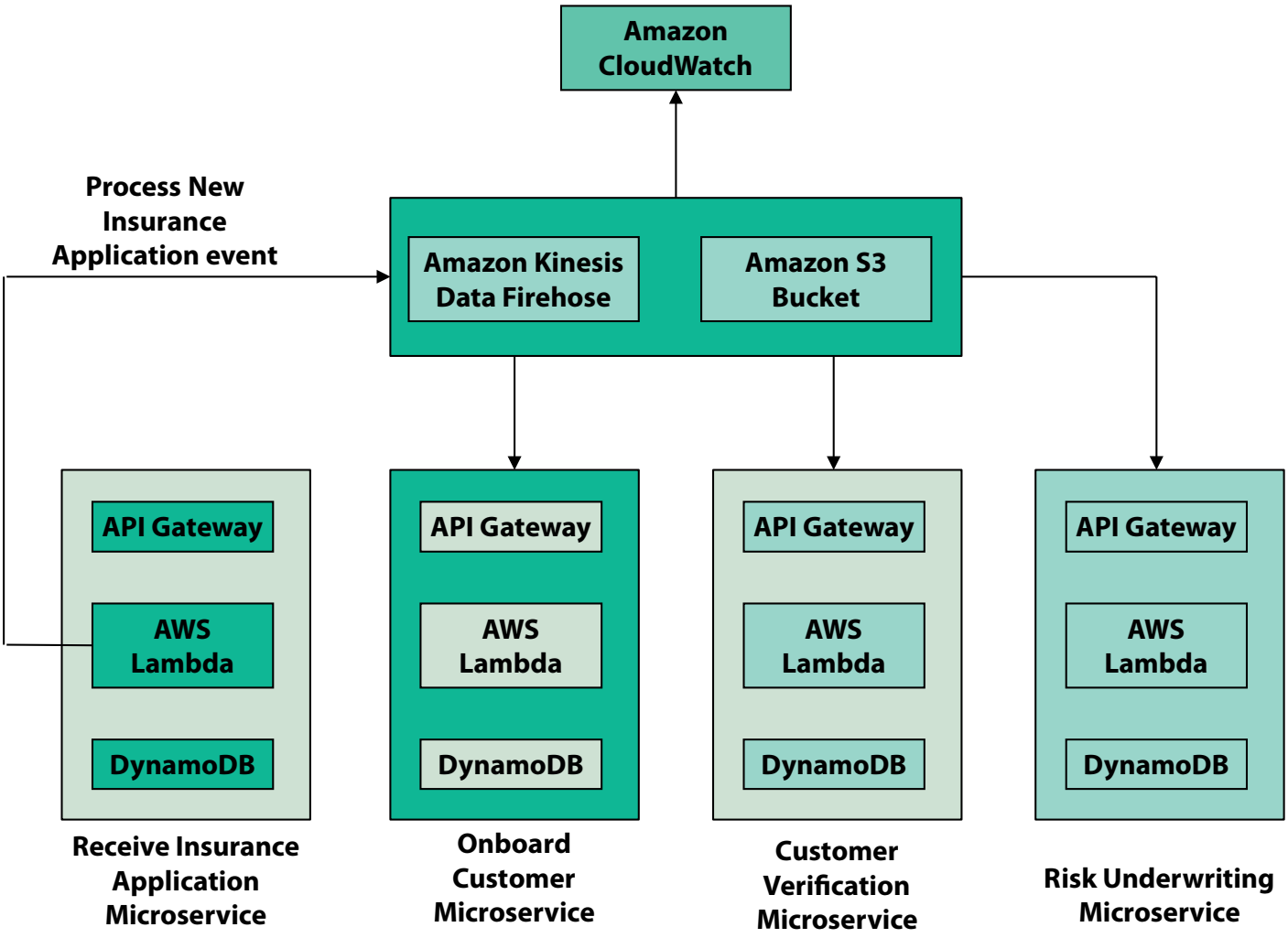


Figure 8: Event Driven Microservices with Streaming on Amazon AWS

Conclusion

Combination of microservices with event driven architecture and streaming is going to be the DNA of live enterprises. It is possible to design super scalable, fault tolerant, high performant & mission critical systems using event driven microservices on streaming backbone. This necessitates architects to address design concerns such as distributed transaction management & exactly once message delivery semantics. But technology has greatly advanced to provide sophisticated processing engines & frameworks. We considered example architectures based on Apache Kafka, Apache Samza, Apache Flink & Amazon Web Services. This can guide architects to implement event driven microservices by leveraging various streaming frameworks.

Financial institutions can overcome classic ETL (Extract – Transform – Load) limitations & latency problems & build highly scalable real-time data integration flows that can stream data to the last mile.

About the Author

Niranjan Kulkarni
Senior Technology Architect, Financial Services, Infosys Limited

Niranjan enjoys working in Enterprise Architecture, Cloud Native Microservices and Java ecosystem. He has over 17 years of experience in Technology Architecture, Solution Design & Delivery. He loves reading and talking on blockchains. He can be reached at Niranjan_Kulkarni@Infosys.com

Author acknowledges

Author acknowledges that the image (ASLV Launch) on the title page is taken from https://upload.wikimedia.org/wikipedia/en/5/53/ASLV_Launch.jpeg under license as per Creative Commons — Attribution-ShareAlike 3.0 Unported — CC BY-SA 3.0