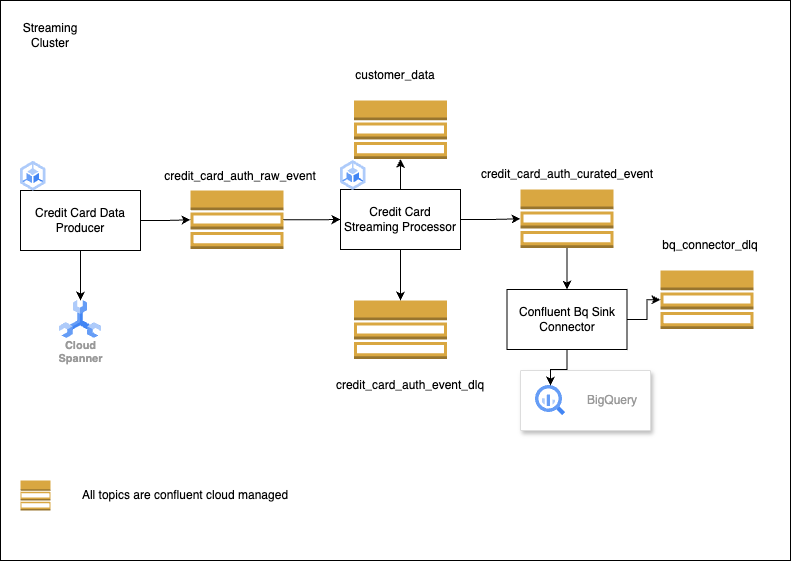
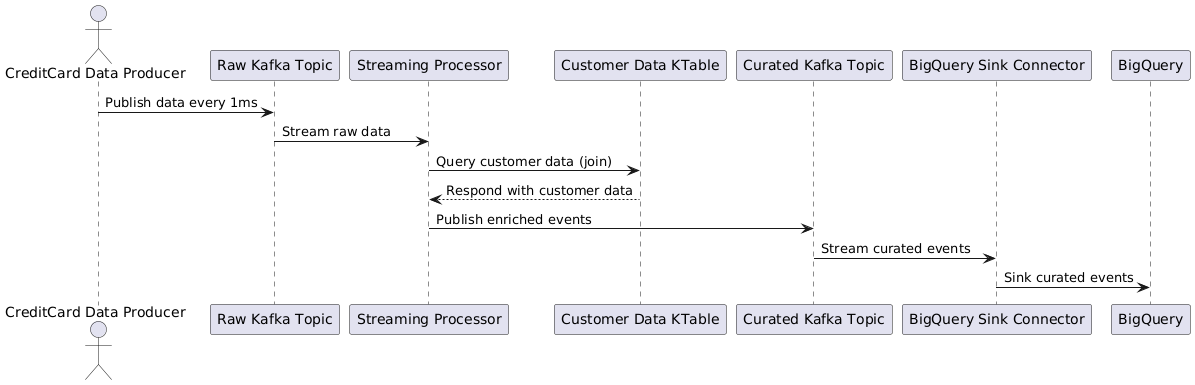
To address the case study requirements for ABC Bank’s streaming solution using Apache Kafka, we can break down the design into several key components. Here’s an end-to-end approach:

1. **High-Level Architecture**



**Low Level Design:**



The architecture consists of multiple layers, each responsible for a different function in the overall pipeline. These layers include:

Source Layer: Create an scheduler to generate raw data in regular intervals (Credit Card Data Generator)

Data Ingestion Layer: Ingests the credit card authorization events. (Kafka Source Topic - credit\_card\_auth\_raw\_events )

Transformation & Data Publishing Layer: Filters PII/PCI fields and enriches the data using customer details and Publishes enriched events for downstream systems.(Kafka Destination Topic - credit\_card\_auth\_curated\_events). SpringBoot Application

Storage Layer: Persists curated events into an event store.

**2. Components of the Streaming Solution**

2.1 Ingestion Layer

Kafka Topics: Use a Kafka topic like credit\_card\_auth\_raw\_events to ingest raw credit card authorization events. These events are published from third-party sources.

Data Serialization: Raw events are ingested in JSON format. Avro or ProtoBuf can be considered for efficiency depending on the volume.

Example of a Raw Event (JSON):

1. {

2. "cardholderName": "John Doe",

3. "cardholderEmail": "john.doe@example.com",

4. "cardNumber": "1234567812345678",

5. "cardNetwork": "Visa",

6. "cardExpiryDate": "08-25",

7. "billingPostcode": "12345",

8. "cardholderPAN": "AAAAA1234A",

9. "transactionID": "550e8400-e29b-41d4-a716-446655440000",

10. "transactionTimestamp": "2024-09-15 12:35:45.123",

11. "transactionType": "A",

12. "txnAmount": 100.50,

13. "txnFee": 2.50,

14. "txnCurrency": "USD"

15. }

16.

2.2 Transformation Layer

Data Filtering: Remove sensitive PII/PCI data such as cardNumber, and use customerPAN to join customer data.

Data Enrichment: Enrich the event with customer details like customerName, customerAddress, DOB, etc., by querying a Customer Data Platform (CDP).

Customer Data Schema:

1. {

2. "customerName": "John Doe",

3. "customerPAN": "AAAAA1234A",

4. "customerAddress": "123 Main St, City, State, ZIP",

5. "customerDOB": "1985-06-15",

6. "customerGender": "Male",

7. "monthlyIncome": 5000.00

8. }

9.

Transformation Pipeline:

Fetch customer information using the customerPAN as a foreign key.

Remove PII fields such as cardNumber.

Retain necessary details like transactionID, txnAmount, txnCurrency, and add customer data.

Example of a Curated Enriched Event (JSON):

1. {

2. "cardholderName": "John Doe",

3. "cardholderEmail": "john.doe@example.com",

4. "cardNetwork": "Visa",

5. "cardExpiryDate": "08-25",

6. "billingPostcode": "12345",

7. "transactionID": "550e8400-e29b-41d4-a716-446655440000",

8. "transactionTimestamp": "2024-09-15 12:35:45.123",

9. "transactionType": "A",

10. "txnAmount": 100.50,

11. "txnFee": 2.50,

12. "txnCurrency": "USD",

13. "customerName": "John Doe",

14. "customerAddress": "123 Main St, City, State, ZIP",

15. "customerDOB": "1985-06-15",

16. "customerGender": "Male",

17. "monthlyIncome": 5000.00

18. }

19.

2.3 Publishing Layer

Kafka Topics: Curated events are published to a Kafka topic such as credit\_card\_auth\_curated\_events. Consumers like fraud detection, analytics, or other business systems can subscribe to this topic.

API for Consumption: Build a REST API service that allows downstream systems to query the curated data in real-time. This API could be backed by a Kafka consumer that fetches events from the credit\_card\_auth\_curated\_events topic or reads from the event store.

2.4 Persistence Layer

Event Store: The curated events are persisted into a cloud-native event store like Google Cloud Storage or BigQuery for long-term storage and analytics. The data can also be persisted in a document database like MongoDB or Elasticsearch to enable real-time access via API.

Data Serialization: Persist events in Avro or ProtoBuf format to optimize storage and retrieval.

**3. Data Flow Example**

Ingestion: Third-party systems send raw credit card authorization events to the credit\_card\_auth\_raw\_events Kafka topic.

Processing and Enrichment:

A Kafka Stream application subscribes to the credit\_card\_auth\_raw\_events topic.

It filters out PII/PCI fields like cardNumber.

It enriches the data by querying customer data using the customerPAN.

It publishes the enriched and transformed events to the credit\_card\_auth\_curated\_events Kafka topic.

Publishing:

Downstream systems (like fraud detection) subscribe to the credit\_card\_auth\_curated\_events topic.

API services can fetch real-time data from Kafka or the event store.

**4. Technologies and Tools**

Kafka (Apache Kafka on GCP): Used for real-time ingestion, transformation, and publication of credit card authorization events.

Kafka Streams: Used to process and transform the events leveraging springboot framework.

Google Cloud Platform (GCP): As the target public cloud, components such as Google Kubernetes Engine, Google Cloud Storage, or BigQuery or Spanner can be used for storage and access.

Avro/ProtoBuf: Efficient serialization format for event storage and API consumption.

**5. Business Value and Use Cases**

By integrating third-party credit card systems with ABC Bank’s cloud platform, the bank can:

Fraud Detection: Enable real-time fraud detection by monitoring enriched events.

Customer Insights: Improve analytics and insights based on real-time credit card usage.

Real-Time Use Cases: Enable other business use cases like personalized offers or notifications based on transaction data.

**6. Deployment and Testing**

Local Setup: Use Docker to set up a local Kafka environment, including Kafka brokers, Zookeeper, and Kafka Connect.

Google Cloud Deployment: Deploy Kafka Stream applications on GCP using Google Kubernetes Engine (GKE) or Cloud Run for scaling, and use Confluent Kafka for kafka broker.

Monitoring: Use Prometheus/Grafana to monitor Kafka health and performance.(TBD)