**Implementation** of **Real-Time Credit Card Fraud Detection System**:

* **Apache NiFi** (data generation & Kafka producer)
* **Azure Event Hubs** (Kafka-compatible broker)
* **Azure Databricks (PySpark)** (stream processing)
* **Delta Lake on ADLS Gen2** (raw & processed zones)
* **Great Expectations** (data quality validation)
* **Unity Catalog** (data governance & security)
* **Power BI** (real-time dashboard)

**1. Apache NiFi – Real-Time Data Generator & Kafka Publisher**

**NiFi Flow Design:**

nginx

GenerateFlowFile → UpdateAttribute → ReplaceText → PutKafkaRecord\_2\_0

**ReplaceText → Simulate Transaction JSON:**

{

"transactionId": "TX${UUID()}",

"cardNumber": "9876-XXXX-XXXX-${random():substring(0,4)}",

"amount": ${math.random():multiply(200000):round()},

"location": "${location:random()}",

"timestamp": "${now():format('yyyy-MM-dd''T''HH:mm:ss''Z')}",

"userId": "U${random():mod(200):plus(100)}"

}

Add this in **Advanced User-Defined Properties**:

location : ${literal("Delhi,London,Mumbai,NYC,Bangalore"):split(","):get(${math.random():mod(5)})}

**Kafka Configuration for PutKafkaRecord\_2\_0:**

Topic Name: transactions

Bootstrap Servers: <eventhub-namespace>.servicebus.windows.net:9093

security.protocol: SASL\_SSL

sasl.mechanism: PLAIN

sasl.jaas.config: org.apache.kafka.common.security.plain.PlainLoginModule required username="$ConnectionString" password="<your-connection-string>";

**2. Azure Event Hubs (Kafka-Compatible)**

* **Namespace**: fraud-detection-ns
* **Event Hub (topic)**: transactions
* Enable Kafka protocol
* Assign Shared Access Policy for NiFi

**3. Databricks Streaming (PySpark) with Great Expectations**

**Schema Definition:**

from pyspark.sql.types import \*

schema = StructType([

StructField("transactionId", StringType()),

StructField("cardNumber", StringType()),

StructField("amount", DoubleType()),

StructField("location", StringType()),

StructField("timestamp", StringType()),

StructField("userId", StringType())

])

**Read Kafka Stream:**

df\_kafka = spark.readStream \

.format("kafka") \

.option("kafka.bootstrap.servers", "<eventhub-namespace>.servicebus.windows.net:9093") \

.option("subscribe", "transactions") \

.option("kafka.security.protocol", "SASL\_SSL") \

.option("kafka.sasl.mechanism", "PLAIN") \

.option("kafka.sasl.jaas.config", "org.apache.kafka.common.security.plain.PlainLoginModule required username=\"$ConnectionString\" password=\"<your-connection-string>\";") \

.load()

from pyspark.sql.functions import \*

df\_json = df\_kafka.selectExpr("CAST(value AS STRING)") \

.withColumn("data", from\_json(col("value"), schema)) \

.select("data.\*")

**4. Great Expectations Validation**

**Wrap and Validate with GE:**

from great\_expectations.dataset import SparkDFDataset

ge\_df = SparkDFDataset(df\_json)

ge\_df.expect\_column\_values\_to\_not\_be\_null("transactionId")

ge\_df.expect\_column\_values\_to\_be\_between("amount", 1, 500000)

Add GE validation logic in Databricks notebook or as a separate checkpoint.

**5. Fraud Detection Logic**

df\_flagged = df\_json.withColumn("isFraud", when(

(col("amount") > 75000) | (col("location").isin("London", "NYC")), True

).otherwise(False))

**6. Unity Catalog – Delta Table Registration**

**Create Catalog and Schemas**

CREATE CATALOG fraud\_analytics;

CREATE SCHEMA fraud\_analytics.raw\_layer;

CREATE SCHEMA fraud\_analytics.processed\_layer;

**Stream Writes to Unity Catalog Tables:**

df\_json.writeStream \

.format("delta") \

.option("checkpointLocation", "/mnt/checkpoints/raw") \

.table("fraud\_analytics.raw\_layer.raw\_transactions")

df\_flagged.writeStream \

.format("delta") \

.option("checkpointLocation", "/mnt/checkpoints/processed") \

.table("fraud\_analytics.processed\_layer.transactions\_processed")

**7. Unity Catalog Security Setup**

**Masking & Tagging:**

-- Tag PII fields

CREATE TAG pii;

ALTER TABLE fraud\_analytics.processed\_layer.transactions\_processed

SET TAG pii ON COLUMN cardNumber;

-- Apply masking

CREATE MASKING POLICY mask\_card\_number

AS (val STRING) -> STRING

RETURN CONCAT('XXXX-XXXX-XXXX-', RIGHT(val, 4));

ALTER TABLE fraud\_analytics.processed\_layer.transactions\_processed

ALTER COLUMN cardNumber SET MASKING POLICY mask\_card\_number;

**Access Controls:**

GRANT SELECT ON TABLE fraud\_analytics.processed\_layer.transactions\_processed TO `bi\_team`;

**8. Power BI – Real-Time Dashboard**

**Power BI + Databricks SQL Connector**

* Use fraud\_analytics.processed\_layer.transactions\_processed as source.
* Create visuals:
  + Fraud vs Non-Fraud trend
  + Map: Top fraud-prone locations
  + User-wise transaction timeline

**Summary of Resources You’ll Have**

| **Resource** | **Description** |
| --- | --- |
| Apache NiFi | Flow to simulate and publish transactions |
| Azure Event Hubs | Kafka ingestion for real-time data |
| Databricks PySpark Notebook | Streaming + Fraud Detection + GE |
| Great Expectations Suite | Expectations for schema and field rules |
| Unity Catalog | Governance, access, masking, lineage |
| Power BI | Real-time dashboards with fraud insights |