**Real-Time Streaming Pipeline in Azure (Kafka + Spark + Delta)**

| **Component** | **Purpose** |
| --- | --- |
| **Azure Event Hubs (Kafka-compatible)** | Acts as Kafka message broker |
| **Azure Databricks (PySpark)** | Reads & processes Kafka data |
| **Python Kafka Producer (on Azure VM)** | Sends dummy data to Event Hub |
| **Azure Data Lake / Delta Lake** | Stores streaming output for analysis |

**WORKFLOW**

**1. Create Azure Event Hubs (Kafka-compatible)**

1. Go to Azure Portal → Search **Event Hubs** → Create:
   * Namespace: your-kafka-namespace
   * Pricing Tier: **Standard**
   * Enable Kafka protocol (enabled by default)
2. Inside the namespace → Create Event Hub:
   * Name: transactions
3. Go to “Shared Access Policies” → Click **RootManageSharedAccessKey**
   * Copy the **Connection String – Primary Key**

Keep:

* EVENT\_HUB\_NAMESPACE → your-kafka-namespace.servicebus.windows.net
* TOPIC → transactions
* CONNECTION\_STRING → Event Hub Shared Access connection string

**2. Provision Azure VM to Run Kafka Producer**

1. Go to Azure Portal → Create **Ubuntu VM**
2. Install Python:

sudo apt update

sudo apt install python3 python3-pip -y

pip3 install kafka-python

1. Create Python file producer.py:

import json, random, time

from datetime import datetime

from kafka import KafkaProducer

producer = KafkaProducer(

bootstrap\_servers='<EVENT\_HUB\_NAMESPACE>:9093',

security\_protocol='SASL\_SSL',

sasl\_mechanism='PLAIN',

sasl\_plain\_username='$ConnectionString',

sasl\_plain\_password='<CONNECTION\_STRING>',

value\_serializer=lambda v: json.dumps(v).encode('utf-8'),

key\_serializer=lambda k: k.encode('utf-8')

)

users = ["U100", "U101", "U102"]

locations = ["Mumbai", "Delhi", "Bangalore", "NYC", "London"]

def generate\_txn():

return {

"transactionId": f"TX{random.randint(1000,9999)}",

"cardNumber": f"9876-XXXX-XXXX-{random.randint(1000,9999)}",

"amount": round(random.uniform(100, 100000), 2),

"location": random.choice(locations),

"timestamp": datetime.utcnow().isoformat(),

"userId": random.choice(users)

}

while True:

txn = generate\_txn()

print("Sending:", txn)

producer.send("transactions", key=txn["transactionId"], value=txn)

time.sleep(1)

1. Run the script:

python3 producer.py

**3. Databricks: Read Kafka Stream via PySpark**

**In a Databricks notebook:**

from pyspark.sql.functions import from\_json, col

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())

])

event\_hub\_config = {

"kafka.bootstrap.servers": "<EVENT\_HUB\_NAMESPACE>:9093",

"subscribe": "transactions",

"startingOffsets": "latest",

"kafka.security.protocol": "SASL\_SSL",

"kafka.sasl.mechanism": "PLAIN",

"kafka.sasl.jaas.config": 'org.apache.kafka.common.security.plain.PlainLoginModule required username="$ConnectionString" password="<CONNECTION\_STRING>";'

}

df\_kafka = spark.readStream \

.format("kafka") \

.options(\*\*event\_hub\_config) \

.load()

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

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

.select("data.\*")

df\_json.display()

**4. Write to Delta Lake (optional)**

Mount ADLS Gen2 to Databricks if not already done.

df\_json.writeStream \

.format("delta") \

.outputMode("append") \

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

.start("/mnt/delta/transactions")

**Final Architecture**

