Users Streaming Data Ingestion Using Spark

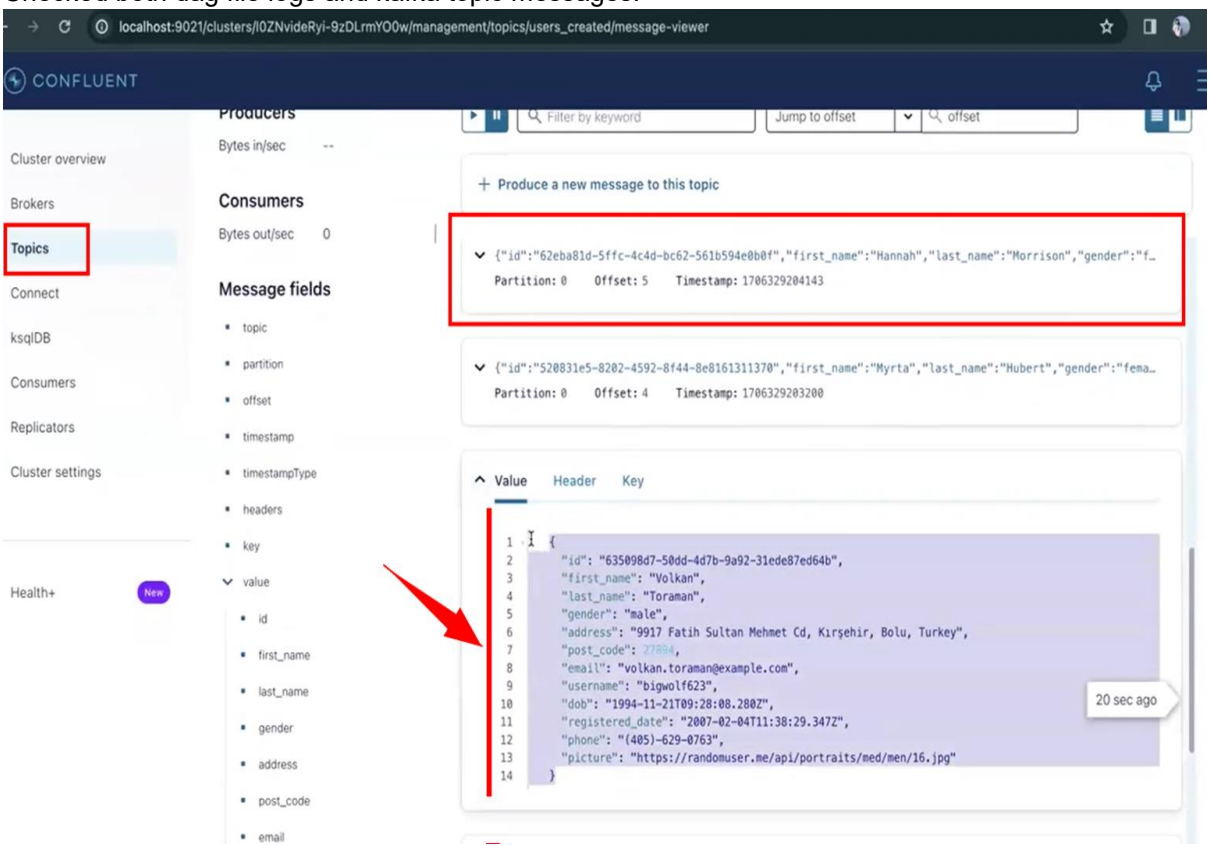
1. installed confluent kafka & airflow locally following the doc notes. Opened both kafka & airflow dashboard on browser locally

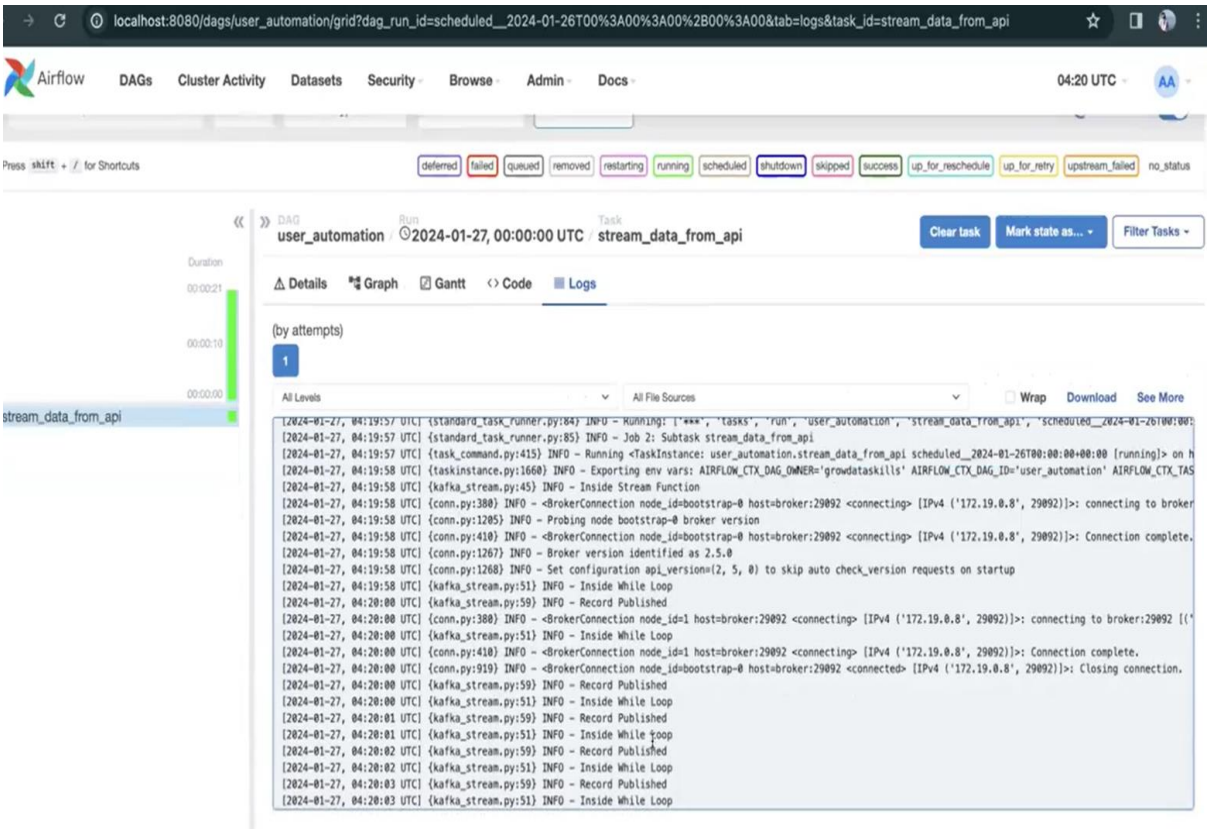
2. we were publishing data to kafka topic using the airflow script

```
def stream_data():
    import json
    from kafka import KafkaProducer
    import time
    import logging

    #create producer object
    producer = KafkaProducer(bootstrap_servers=['broker:29092'], max_block_ms=5000)
    curr_time = time.time()

    while True:
        if time.time() > curr_time + 300: #1 minute
            break
        try:
            res = get_data()
            res = format_data(res)

            producer.send('users_data', json.dumps(res).encode('utf-8'))
            print("Record Inserted !!")
        except Exception as e:
            logging.error(f'An error occured: {e}')
            continue


with DAG('user_automation',
         default_args=default_args,
         schedule_interval='@daily',
         catchup=False) as dag:

    streaming_task = PythonOperator(
        task_id='stream_data_from_api',
        python_callable=stream_data
    )
```

3. so we create kafka topic 'users_data' in confluent kafka and schedule the dag file where records started publishing to kafka topic.

   Checked both dag file logs and kafka topic messages:





4. Next the data was streaming from kafka and ingesting into cassandra. For this wrote a spark_stream script where first I established a spark connection. Inside spark connection we included all the jar files using which spark can communicate with both kafka and cassandra. Used jar files based on our pyspark version.

```
def create_spark_connection():
    s_conn = None

    try:
        print("Creating spark session!")
        print("*****************************************************")
        s_conn = SparkSession.builder \
            .appName('SparkDataStreaming') \
            .config('spark.jars.packages', "com.datastax.spark:spark-cassandra-connector_2.12:3.4.0,"
                                           "org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.0") \
            .config('spark.cassandra.connection.host', 'localhost') \
            .getOrCreate()

        s_conn.sparkContext.setLogLevel("ERROR")
        print("Spark connection created successfully!")
        print("*****************************************************")
    except Exception as e:
        print(f"Couldn't create the spark session due to exception {e}")
        print("*****************************************************")

    return s_conn
```

5. Once spark connection is done, next we created dataframe by connecting & reading data from kafka topic. So we were getting data in the form of key value pair where we were using custom schema to get a clean row columnar based dataframe.

```
def connect_to_kafka(spark_conn):
    spark_df = None
    try:
        spark_df = spark_conn.readStream \
            .format('kafka') \
            .option('kafka.bootstrap.servers', 'localhost:9092') \
            .option('subscribe', 'users_data') \
            .option('startingOffsets', 'latest') \
            .load()

        print("kafka dataframe created successfully")
        print("*****************************************************")
    except Exception as e:
        print(f"kafka dataframe could not be created because: {e}")
        print("*****************************************************")

    return spark_df

def create_selection_df_from_kafka(spark_df):
    schema = StructType([
        StructField("id", StringType(), False),
        StructField("first_name", StringType(), False),
        StructField("last_name", StringType(), False),
        StructField("gender", StringType(), False),
        StructField("address", StringType(), False),
        StructField("post_code", StringType(), False),
        StructField("email", StringType(), False),
        StructField("username", StringType(), False),
        StructField("registered_date", StringType(), False),
        StructField("phone", StringType(), False),
        StructField("picture", StringType(), False)
    ])

    sel = spark_df.selectExpr("CAST(value AS STRING)") \
        .select(from_json(col('value'), schema).alias('data')).select("data.*")

    return sel
```

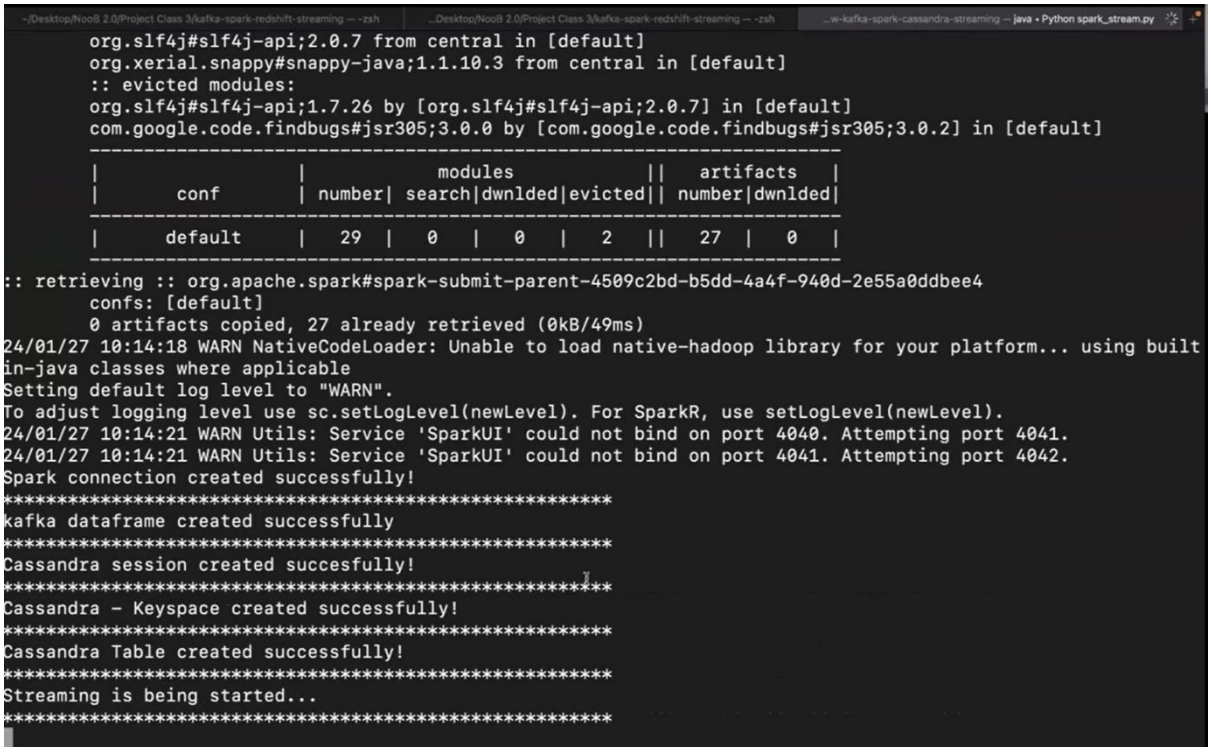
6. Next we created cassandra connection. We created keyspace and table using connection.

```python
def create_spark_connection():
    s_conn = None

    try:
        print("Creating spark session!")
        print("*****************************************************")
        s_conn = SparkSession.builder \
            .appName('SparkDataStreaming') \
            .config('spark.jars.packages', "com.datastax.spark:spark-cassandra-connector_2.12:3.4.0,"
                                           "org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.0") \
            .config('spark.cassandra.connection.host', 'localhost') \
            .getOrCreate()

        s_conn.sparkContext.setLogLevel("ERROR")
        print("Spark connection created successfully!")
        print("*****************************************************")
    except Exception as e:
        print(f"Couldn't create the spark session due to exception {e}")
        print("*****************************************************")

    return s_conn

def create_keyspace(session):
    session.execute("""
        CREATE KEYSPACE IF NOT EXISTS spark_streams
        WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'};
    """)

    print("Cassandra - Keyspace created successfully!")
    print("*****************************************************")


def create_table(session):
    session.execute("""
    CREATE TABLE IF NOT EXISTS spark_streams.created_users (
        id TEXT PRIMARY KEY,
        first_name TEXT,
        last_name TEXT,
        gender TEXT,
        address TEXT,
        post_code TEXT,
        email TEXT,
        username TEXT,
        registered_date TEXT,
        phone TEXT,
        picture TEXT);
    """)

    print("Cassandra Table created successfully!")
    print("*****************************************************")
```

7. Then we were writing dataframe into cassandra using spark writeStream

```python
def foreach_batch_function(df, epoch_id):
    # Print each micro-batch for debugging
    print("Records inserted..")
    print("\n")
    print(f"Batch {epoch_id}")
    df.show()
    print("*****************************************************")

if __name__ == "__main__":
    # create spark connection
    spark_conn = create_spark_connection()

    if spark_conn is not None:
        # connect to kafka with spark connection
        spark_df = connect_to_kafka(spark_conn)
        selection_df = create_selection_df_from_kafka(spark_df)
        session = create_cassandra_connection()

        if session is not None:
            create_keyspace(session)
            create_table(session)

            print("Streaming is being started...")
            print("*****************************************************")

            streaming_query = (selection_df.writeStream.foreachBatch(foreach_batch_function)
                               .outputMode("append")
                               .format("org.apache.spark.sql.cassandra")
                               .option('checkpointLocation', '/tmp/checkpoint')
                               .option('keyspace', 'spark_streams')
                               .option('table', 'created_users')
                               .start())

            streaming_query.awaitTermination()
```

8. Checked spark streaming app running successfully



9. At last, checked Cassandra table records using cql command