Delta Migration from Informix to Postgres

The challenge is to move/migrate delta data from Informix to postgres via kafka. The following main requirements have to be considered.

1. **Create a Postgres Script for Audit Trail**
2. **Producer Update**
3. **Consumer Update.**

1. Detailed Specification

I. Create a Postgres Database Script:

Sql script need to be created with below requirements

a. Create “**schema**” under default “postgres” DB with name **auditlog**
b. Create **producer_log** with below fields
   i. SEQ_ID **(PK)**
   ii. TOPICNAME
   iii. SOURCE
   iv. SHCEMA_NAME
   v. TABLE_NAME
   vi. PRODUCER_PAYLOAD
   vii. OPERATION
c. Create **consumer_log** with below fields
   i. SEQ_ID (FK)
   ii. TOPICNAME
   iii. DESTINATION
   iv. SCHEMA_NAME
   v. CONSUMAER_QUERY

d. Create **audit_log** with below details
    i. SEQ_ID (FK)
    ii. PRODUCER_PUBLISH_STATUS
    iii. PRODUCER_FAILURE_LOG
    iv. PRODUCER_PUBLISH_TIME
    v. CONSUMER_DEPLOY_STATUS
    vi. CONSUMER_FAILURE_LOG
    vii. CONSUMAER_UPDATE_TIME

II. Producer Server

a) When any DML operation takes place on the Informix table, the DB trigger created on that table will post (via Informix audit script) details of operations type (insert/delete/update) along with table data to node listener via json payload.
b) Configure Express server to consume more than 2MB data with no restriction
c) **Topic** name should get it from environment configuration and updated with payload (Note: Now it was hard coded for development purpose)
d) Create unique sequential ID and integrate with Payload
e) SOURCE database information will be shared in environment. This will be in “INFORMIX” for this requirement
f) Insert the data in producer_log table
g) Post the payload to Kafka server with respective topic which is configured
h) If topic published is success, update the field in audit_log table
    i. SEQ_ID
    ii. PRODUCER_PUBLISH_STATUS
    iii. PRODUCER_FAILURE_LOG (NULL)
    iv. PRODUCER_PUBLISH_TIME

i) If any error occurs on kafka publishing, re-republish the message by incrementing retry counter value to 1. This counter has to be configurable and before re-publishing check maximum retry counter value.
j) After max retry if error persists we need to update the failure with sequence id and error to different kafka topic/queue (lets say ‘db.postgres.error’) with some additional fields like "recipients": ["admin@abc.com”].
k) Also update the audit_log table

III. Consumer need to update data in postgres

l) When any DML operation takes place on the Informix table, the DB trigger created on that table will post (via Informix audit script) details of operations type (insert/delete/update) along with table data to node listener via json payload.
m) This payload posted to Kafka. (node producer processor)
n) DESTINATION database information will be shared in the environment. This will be in “POSTGRESQL” for this requirement.
o) **audit_code**/createtable.sql file on the Informix side has the “Test” table DDL. Ensure that you replicate this table creation at Postgres based on the schema at payload. (use default “tablespace”).

For example :

1) Informix DB → testdb = Default Postgres DB → testdb schema → test table.

2) **For the Audit Trail.**
Default Postgres DB → “**auditlog” schema** → **producer_log**
consumer_log
audit_log tables

p) The consumer processor has to parse this data and mirror or replicate postgres table based on the operation type.
   a. Parse the payload and create query
   b. Insert the consumer_log table based on payload
   c. Then update the Database on respective table
   d. If success, please update the below field on audit_log on respective Sequential ID
      i. CONSUMER_DEPLOY_STATUS
      ii. CONSUMER_FAILURE_LOG
      iii. CONSUMAER_UPDATE_TIME

q) On consumer side if any error occurs handle it. If error is on the postgres DB side, re-republish the message by incrementing retry counter value to 1. This counter has to be configurable and before re-publishing check maximum retry counter value.
r) After max retry if error persists we need to re-publish same payload to different kafka topic/queue (lets say ‘db.postgres.error’) with some additional fields like "recipients": ["admin@abc.com”].
s) Update the audit_log table
t) Figure out implementation approach for above
   a. Blob/Byte (Bytea for postgres) needs to be supported.

IV. Audit Trail (Continued from Req 1).

a) Include audit trail function in new file for both producer and consumer side.
b) Create a table “producer_log”, “ consumer_log” **and** “audit_log” and Capture appropriate data as per field name given below
c) Seq ID is the reference key for all tables.
d) Feel free to include additional fields which are appropriate.
e) Ensure that DB connection pool is taken care of on both producer and consumer side.

2. Environment setup

I. Informix setup:

1) run docker run -p 2021:2021 -it -u root appiriodevops/ifxpg-dbmigration:v1 bash
2) run su informix
3) oninit -vy

II. Kafka Setup

a. open new terminal at your local system.
b. run docker run -itp 9092:9092 appiriodevops/kafka-local:2.12

III. Node Express and producer setup:

a. open new terminal on local machine

b. mkdir /home/<user>/repo
c. cd /home/<user>/repo
d. git clone https://github.com/topcoder-platform/informix-postgres-migrator.git
e. cd /home/<user>/repo/informix-postgres-migrator/ifx_to_pg_challenge/server_setup
f. run npm install
g. run node nodeserver.js
h. make sure nodeserver is running on 8080 port as it is hard coded here - https://github.com/topcoder-platform/informix-postgres-migrator/blob/master/ifx_to_pg_challenge/audit_code/audit_util.c#L454

IV. Node Consumer setup:

a. open new terminal on local machine
b. cd /home/<user>/repo/informix-postgres-migrator/ifx_to_pg_challenge/server_setup
c. run node consumer.js

V. Testing :

a. Connect with Informix server which we are running already #1
b. run cd /home/informix/trunk/dev2
c. run dbaccess -e testdb saveAuditOnFile.sql
d. (Step c, will create a trigger on the table, insert a record, delete trigger and drop table. This is verification for DML update on informix side)
e. This will generate a json file in /tmp. Also it will post the json file to Producer
f. We can see the message in consumer

3. Submission Expectation:

1) Consumer script.
2) Producer script
3) Audit Trail script
4) Verification of all 3 requirements..
5) Documentation

4. Reference tool.

- Kakfa topic is already defined at sample payload.
- Setup local postgres or via docker postgres.
- You can dockerize Informix and postgres using docker-compose.yml and provide integrated solution.