Real-Time Data Streaming from PostgreSQL to Clickhouse Using Kafka

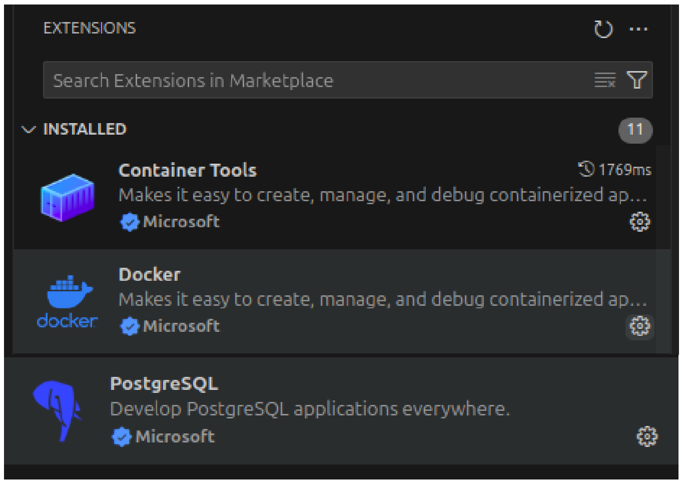
By: Mariam Hossam Goda

Track: Data Engineering Zagazig

My github repo: https://github.com/mariamhossamdiab/Real-Time-Data-Streaming-kafka

---

- ✓ Install extensions



EXTENSIONS

Search Extensions in Marketplace

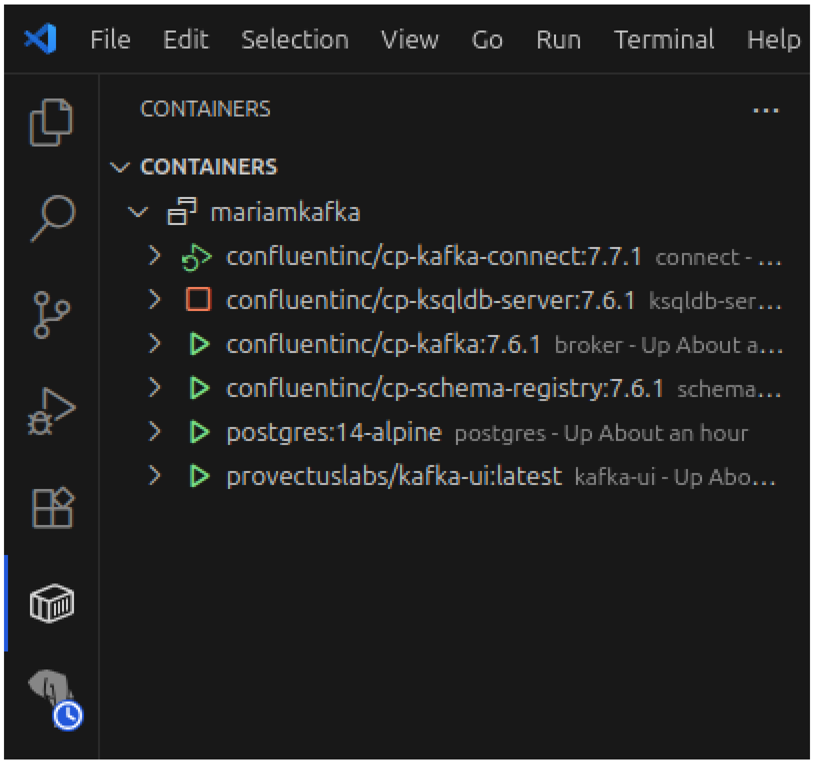
INSTALLED 11

Container Tools 1769ms
Makes it easy to create, manage, and debug containerized ap…
Microsoft

Docker
Makes it easy to create, manage, and debug containerized ap…
Microsoft

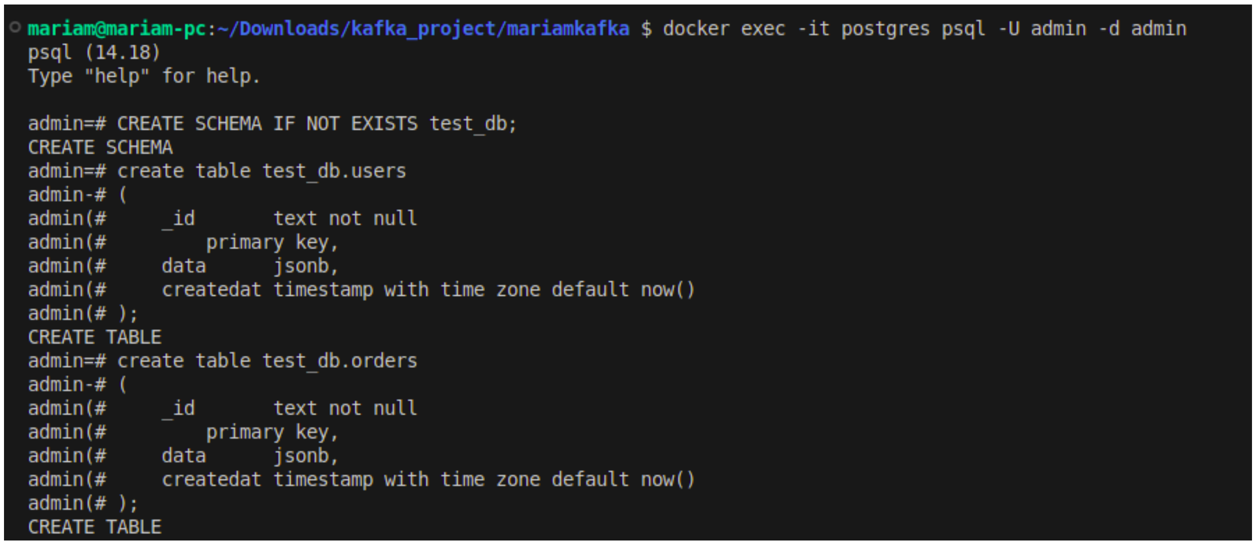
PostgreSQL
Develop PostgreSQL applications everywhere.
Microsoft

- ✓ Running containers of docker compose

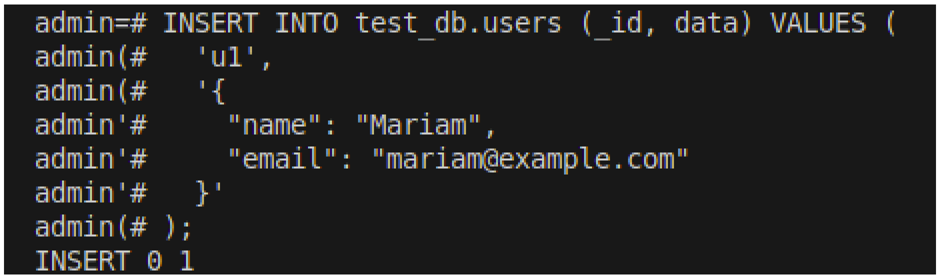


```
confluentinc/cp-kafka-connect:7.7.1  connect - ...
confluentinc/cp-ksqldb-server:7.6.1  ksqldb-ser...
confluentinc/cp-kafka:7.6.1  broker - Up About a...
confluentinc/cp-schema-registry:7.6.1  schema...
postgres:14-alpine  postgres - Up About an hour
provectuslabs/kafka-ui:latest  kafka-ui - Up Abo...
```

Step 1: Creating the Tables in PostgreSQL , I created a new schema called test_db, and added two tables: users and orders. Both use JSONB to hold flexible data. Then, I inserted a few records manually to simulate real transactions.

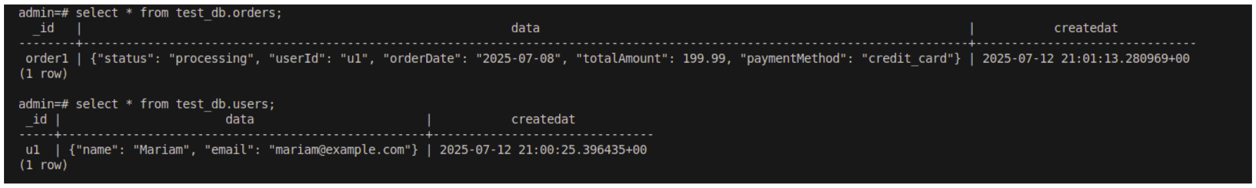


```
mariam@mariam-pc:~/Downloads/kafka_project/mariamkafka $ docker exec -it postgres psql -U admin -d admin
psql (14.18)
Type "help" for help.

admin=# CREATE SCHEMA IF NOT EXISTS test_db;
CREATE SCHEMA
admin=# create table test_db.users
admin-# (
admin(#     _id       text not null
admin(#         primary key,
admin(#     data      jsonb,
admin(#     createdat timestamp with time zone default now()
admin(# );
CREATE TABLE
admin=# create table test_db.orders
admin-# (
admin(#     _id       text not null
admin(#         primary key,
admin(#     data      jsonb,
admin(#     createdat timestamp with time zone default now()
admin(# );
CREATE TABLE
```



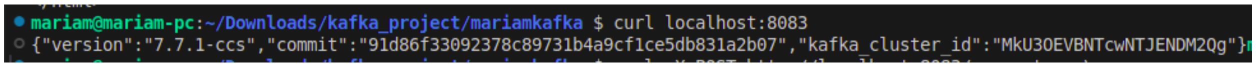
```
admin=# INSERT INTO test_db.users (_id, data) VALUES (
admin(#   'u1',
admin(#   '{
admin'#     "name": "Mariam",
admin'#     "email": "mariam@example.com"
admin'#   }'
admin(# );
INSERT 0 1
```

- ✓ **view records**



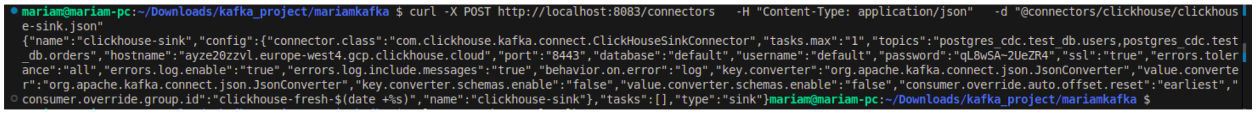
```
admin=# select * from test_db.orders;
  _id   |                                                     data                                                      |          createdat
--------+---------------------------------------------------------------------------------------------------------------+-------------------------------
 order1 | {"status": "processing", "userId": "u1", "orderDate": "2025-07-08", "totalAmount": 199.99, "paymentMethod": "credit_card"} | 2025-07-12 21:01:13.280969+00
(1 row)

admin=# select * from test_db.users;
 _id |                       data                        |           createdat
-----+---------------------------------------------------+-------------------------------
 u1  | {"name": "Mariam", "email": "mariam@example.com"} | 2025-07-12 21:00:25.396435+00
(1 row)
```

- ✓ **health check and an entry point to monitor your Kafka Connect ecosystem**



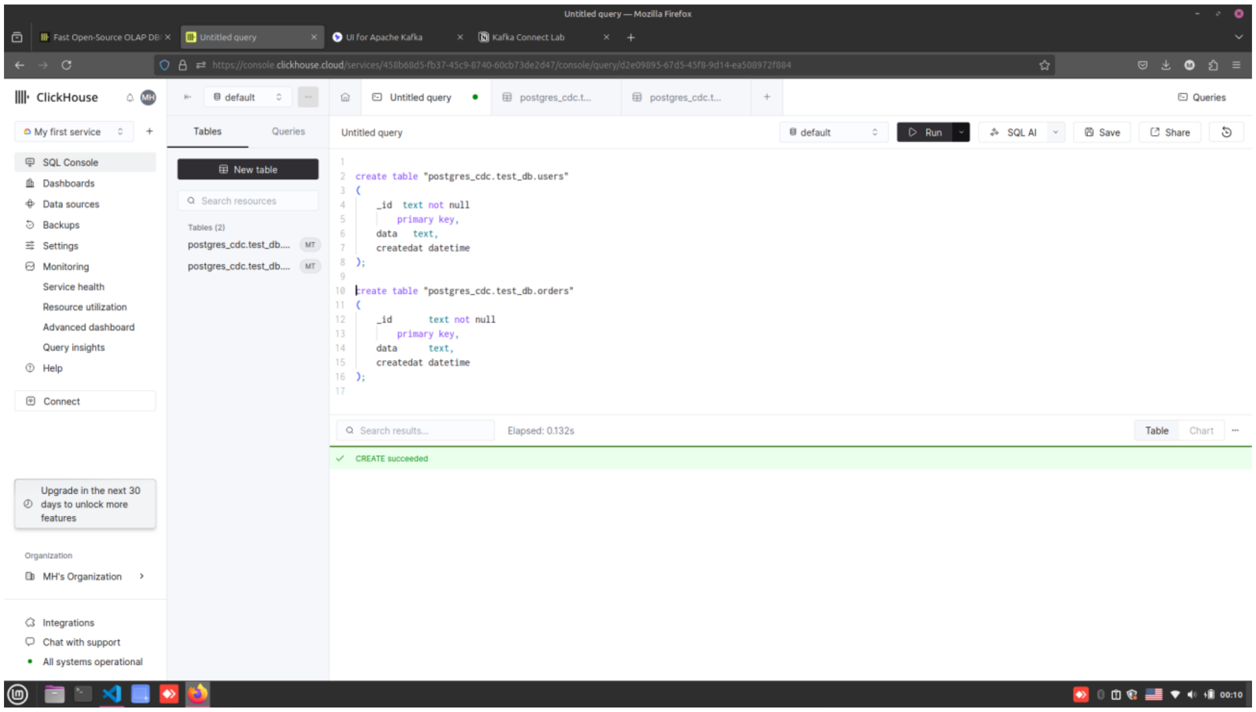
```
mariam@mariam-pc:~/Downloads/kafka_project/mariamkafka $ curl localhost:8083
{"version":"7.7.1-ccs","commit":"91d86f33092378c89731b4a9cf1ce5db831a2b07","kafka_cluster_id":"MkU30EVBNTcwNTJENDM2Qg"}
```

- ✓ **configure a Kafka Connect Sink connector that will transfer data from the specified Kafka topics into a ClickHouse database**



```
mariam@mariam-pc:~/Downloads/kafka_project/mariamkafka $ curl -X POST http://localhost:8083/connectors   -H "Content-Type: application/json"   -d "@connectors/clickhouse/clickhous
e-sink.json"
{"name":"clickhouse-sink","config":{"connector.class":"com.clickhouse.kafka.connect.ClickHouseSinkConnector","tasks.max":"1","topics":"postgres_cdc.test_db.users,postgres_cdc.test
_db.orders","hostname":"ayze20zzvl.europe-west4.gcp.clickhouse.cloud","port":"8443","database":"default","username":"default","password":"qL8wSA~2UeZR4","ssl":"true","errors.toler
ance":"all","errors.log.enable":"true","errors.log.include.messages":"true","behavior.on.error":"log","key.converter":"org.apache.kafka.connect.json.JsonConverter","value.converte
r":"org.apache.kafka.connect.json.JsonConverter","key.converter.schemas.enable":"false","value.converter.schemas.enable":"false","consumer.override.auto.offset.reset":"earliest","
consumer.override.group.id":"clickhouse-fresh-$(date +%s)","name":"clickhouse-sink"},"tasks":[],"type":"sink"}mariam@mariam-pc:~/Downloads/kafka_project/mariamkafka $
```

- ✓ **create tables in cllickhouse**

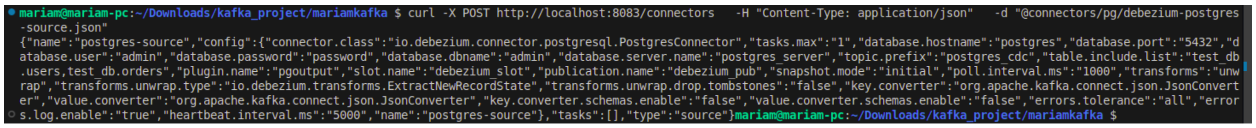


```sql
create table "postgres_cdc.test_db.users"
(
    _id  text not null
        primary key,
    data   text,
    createdat datetime
);

create table "postgres_cdc.test_db.orders"
(
    _id       text not null
        primary key,
    data      text,
    createdat datetime
);
```

Elapsed: 0.132s

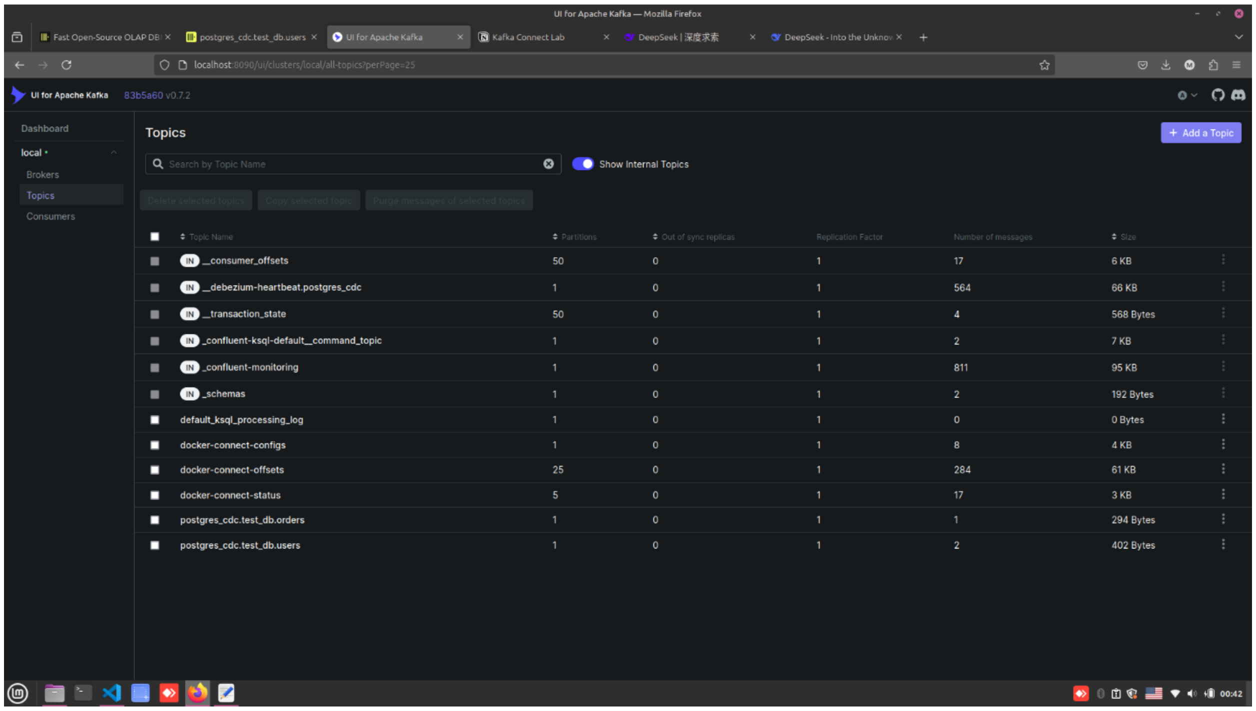
CREATE succeeded

Step 2: Connecting PostgreSQL to Kafka with Debezium I used the JSON config file and used this command to create the connector:



```
mariam@mariam-pc:~/Downloads/kafka_project/mariamkafka $ curl -X POST http://localhost:8083/connectors   -H "Content-Type: application/json"   -d "@connectors/pg/debezium-postgres
-source.json"
{"name":"postgres-source","config":{"connector.class":"io.debezium.connector.postgresql.PostgresConnector","tasks.max":"1","database.hostname":"postgres","database.port":"5432","d
atabase.user":"admin","database.password":"password","database.dbname":"admin","database.server.name":"postgres_server","topic.prefix":"postgres_cdc","table.include.list":"test_db
.users,test_db.orders","plugin.name":"pgoutput","slot.name":"debezium_slot","publication.name":"debezium_pub","snapshot.mode":"initial","poll.interval.ms":"1000","transforms":"unw
rap","transforms.unwrap.type":"io.debezium.transforms.ExtractNewRecordState","transforms.unwrap.drop.tombstones":"false","key.converter":"org.apache.kafka.connect.json.JsonConvert
er","value.converter":"org.apache.kafka.connect.json.JsonConverter","key.converter.schemas.enable":"false","value.converter.schemas.enable":"false","errors.tolerance":"all","error
s.log.enable":"true","heartbeat.interval.ms":"5000","name":"postgres-source"},"tasks":[],"type":"source"}mariam@mariam-pc:~/Downloads/kafka_project/mariamkafka $
```

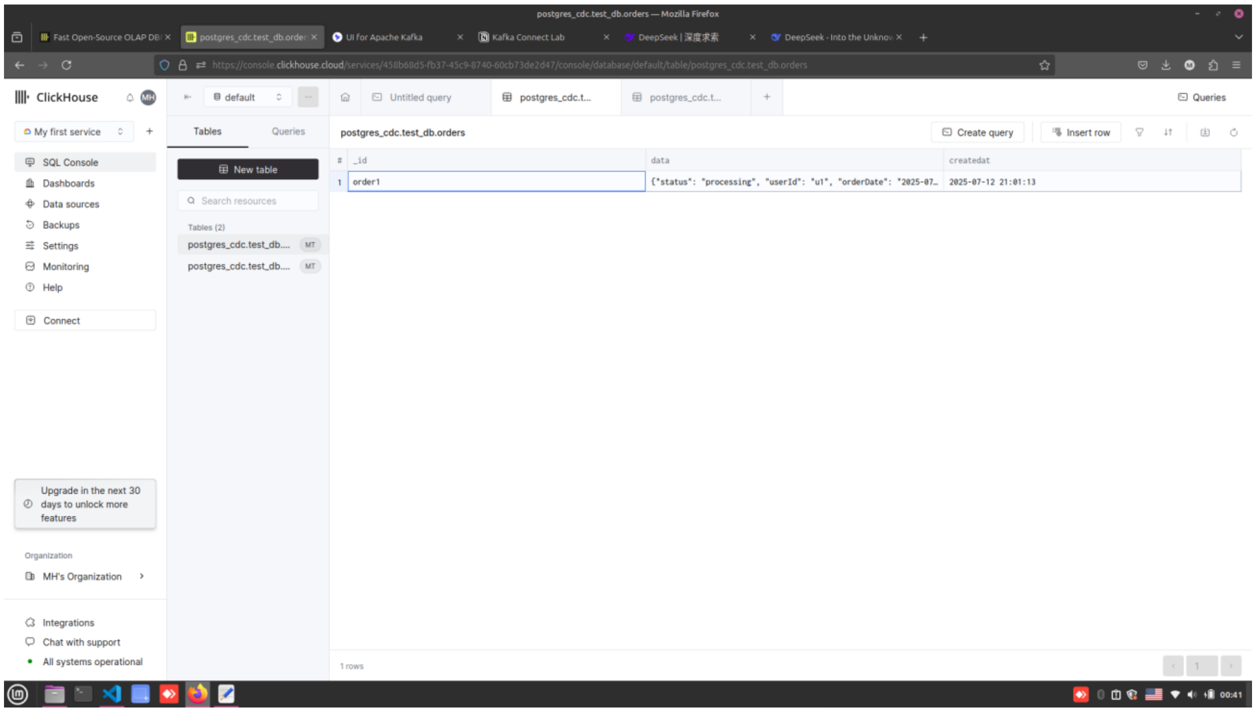
Step 3: Verifying Data in Kafka Topics After the connector was up, I confirmed that Kafka was actually receiving the changes from the database.



Topics

Topic Name	Partitions	Out of sync replicas	Replication Factor	Number of messages	Size
IN __consumer_offsets	50	0	1	17	6 KB
IN __debezium-heartbeat.postgres_cdc	1	0	1	564	66 KB
IN __transaction_state	50	0	1	4	568 Bytes
IN _confluent-ksql-default__command_topic	1	0	1	2	7 KB
IN _confluent-monitoring	1	0	1	811	95 KB
IN _schemas	1	0	1	2	192 Bytes
default_ksql_processing_log	1	0	1	0	0 Bytes
docker-connect-configs	1	0	1	8	4 KB
docker-connect-offsets	25	0	1	284	61 KB
docker-connect-status	5	0	1	17	3 KB
postgres_cdc.test_db.orders	1	0	1	1	294 Bytes
postgres_cdc.test_db.users	1	0	1	2	402 Bytes

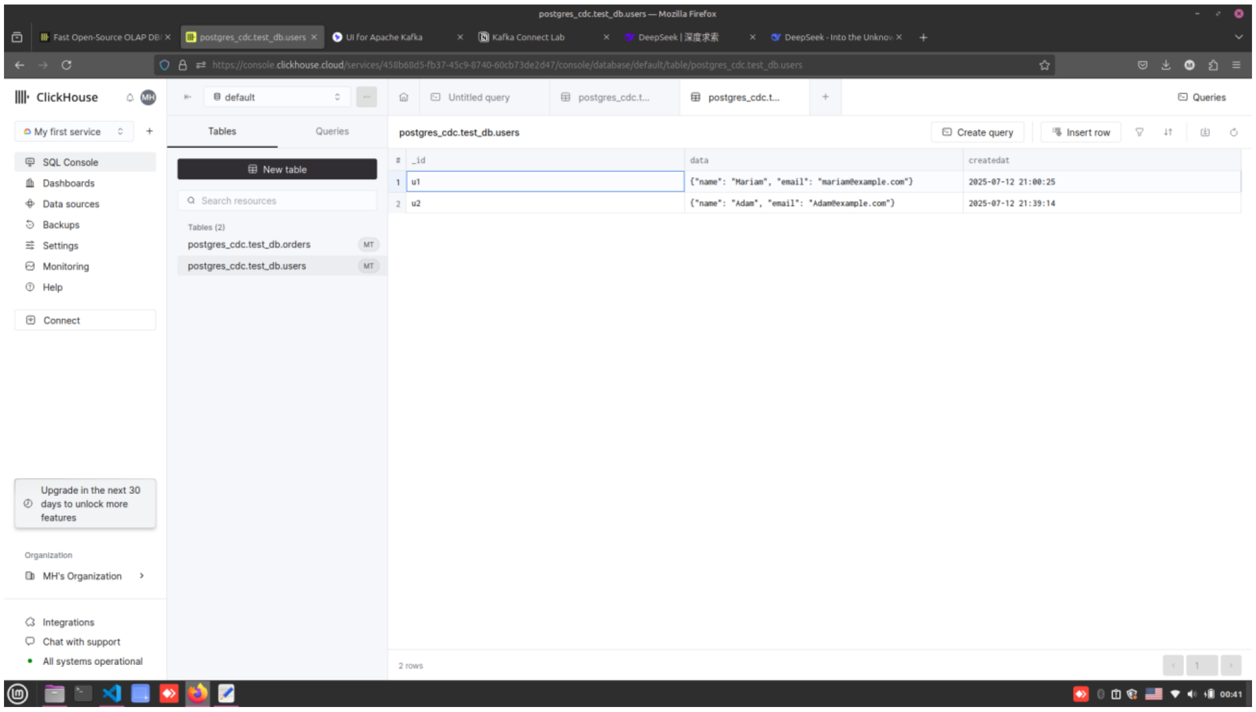
Step 4: Checking the Results in Clickhouse



postgres_cdc.test_db.orders

#	_id	data	createdat
1	order1	{"status": "processing", "userId": "u1", "orderDate": "2025-07…	2025-07-12 21:01:13

1 rows



postgres_cdc.test_db.users

#	_id	data	createdat
1	u1	{"name": "Mariam", "email": "mariam@example.com"}	2025-07-12 21:00:25
2	u2	{"name": "Adam", "email": "Adam@example.com"}	2025-07-12 21:39:14