David Pecollet
Customer Success Architect
david.pecollet@neo4j.com



Rodrigo Haces
Customer Success Architect
rodrigo.haces@neo4j.com



neo4j

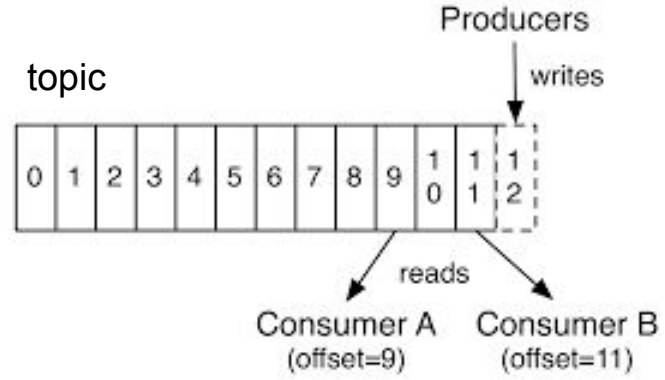

neo4j
neo4j ⇔ kafka
kafka
Integration with neo4j-streams

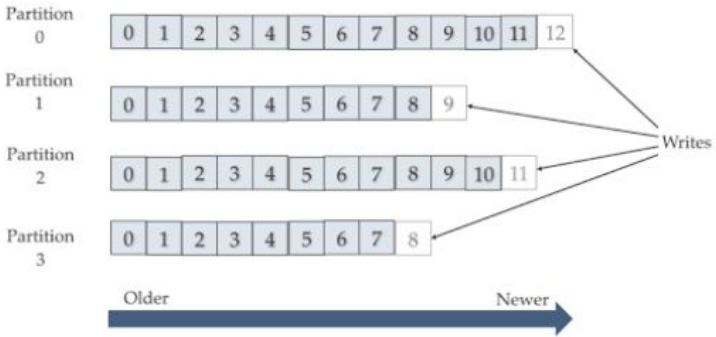
Kafka - Quick Introduction

What it is

- Distributed messaging system
  - fast, scalable
  - persistent

- *Producers* append to topic
- *Consumers* read from topic
  - at consumer-managed offset





What for

- Real-time data feeds between systems
- CDC, database replication

Kafka - Ecosystem

Kafka Connect

Plugins

- Source
- Sink

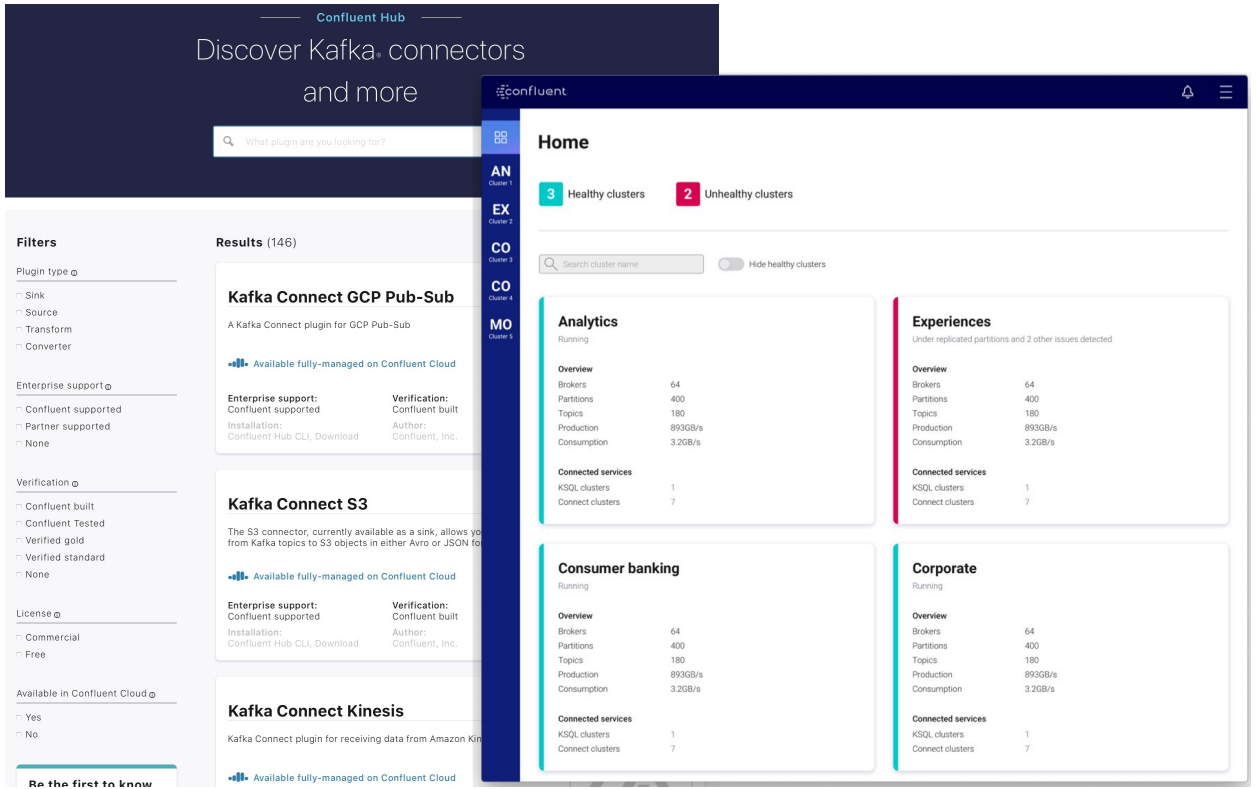
Control Center

GUI

- Monitor
- Configure

KSQL, Kstreams

Stream processing

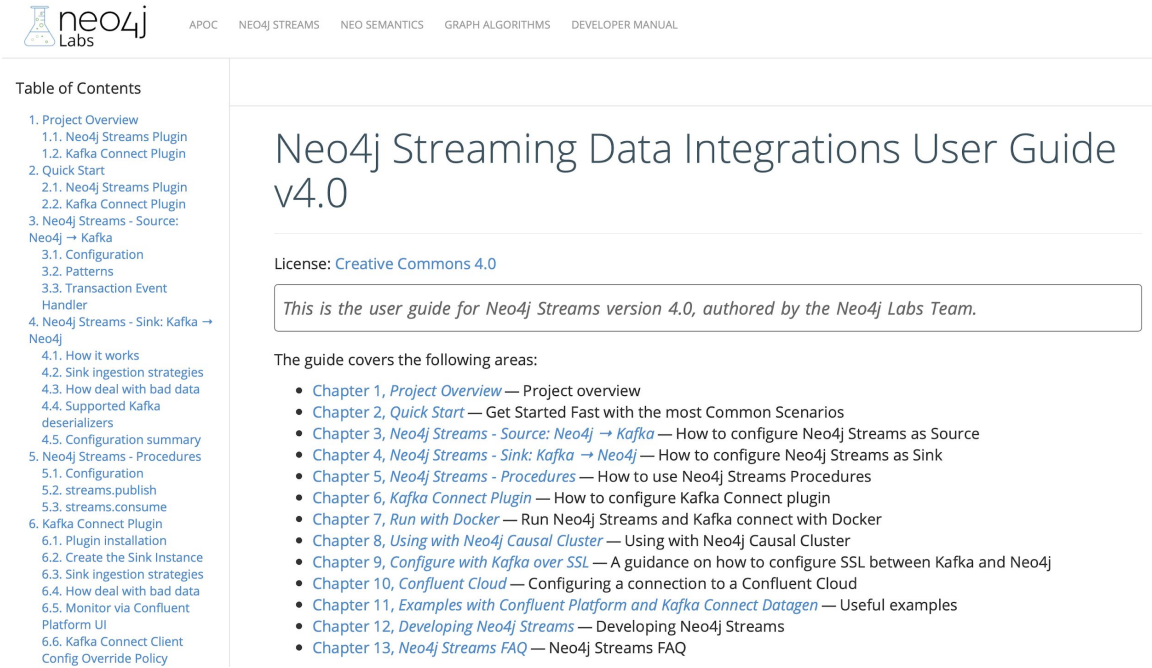


Neo4j Streams

https://github.com/neo4j-contrib/neo4j-streams

https://neo4j.com/docs/labs/neo4j-streams/current/





neo4j Labs

APOC NEO4J STREAMS NEO SEMANTICS GRAPH ALGORITHMS DEVELOPER MANUAL

Table of Contents

1. Project Overview
   1.1. Neo4j Streams Plugin
   1.2. Kafka Connect Plugin
2. Quick Start
   2.1. Neo4j Streams Plugin
   2.2. Kafka Connect Plugin
3. Neo4j Streams - Source: Neo4j → Kafka
   3.1. Configuration
   3.2. Patterns
   3.3. Transaction Event Handler
4. Neo4j Streams - Sink: Kafka → Neo4j
   4.1. How it works
   4.2. Sink ingestion strategies
   4.3. How deal with bad data
   4.4. Supported Kafka deserializers
   4.5. Configuration summary
5. Neo4j Streams - Procedures
   5.1. Configuration
   5.2. streams.publish
   5.3. streams.consume
6. Kafka Connect Plugin
   6.1. Plugin installation
   6.2. Create the Sink Instance
   6.3. Sink ingestion strategies
   6.4. How deal with bad data
   6.5. Monitor via Confluent Platform UI
   6.6. Kafka Connect Client Config Override Policy

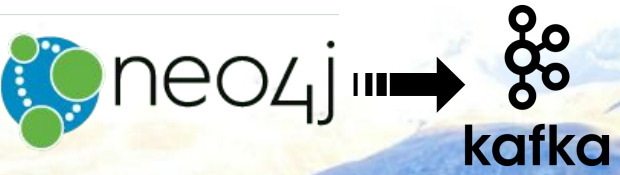
Neo4j Streaming Data Integrations User Guide v4.0

License: Creative Commons 4.0

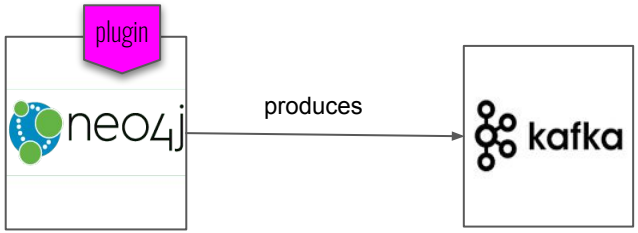
This is the user guide for Neo4j Streams version 4.0, authored by the Neo4j Labs Team.

The guide covers the following areas:

- Chapter 1, *Project Overview* — Project overview
- Chapter 2, *Quick Start* — Get Started Fast with the most Common Scenarios
- Chapter 3, *Neo4j Streams - Source: Neo4j → Kafka* — How to configure Neo4j Streams as Source
- Chapter 4, *Neo4j Streams - Sink: Kafka → Neo4j* — How to configure Neo4j Streams as Sink
- Chapter 5, *Neo4j Streams - Procedures* — How to use Neo4j Streams Procedures
- Chapter 6, *Kafka Connect Plugin* — How to configure Kafka Connect plugin
- Chapter 7, *Run with Docker* — Run Neo4j Streams and Kafka connect with Docker
- Chapter 8, *Using with Neo4j Causal Cluster* — Using with Neo4j Causal Cluster
- Chapter 9, *Configure with Kafka over SSL* — A guidance on how to configure SSL between Kafka and Neo4j
- Chapter 10, *Confluent Cloud* — Configuring a connection to a Confluent Cloud
- Chapter 11, *Examples with Confluent Platform and Kafka Connect Datagen* — Useful examples
- Chapter 12, *Developing Neo4j Streams* — Developing Neo4j Streams
- Chapter 13, *Neo4j Streams FAQ* — Neo4j Streams FAQ



Neo4j Streams Plugin - Source



- Neo4j transaction event listener
  - Triggers on creates/deletes/updates of nodes/relationships/properties
- Filters/extracts according to provided patterns
- Maps data into standard Debezium event format
- Sends event to kafka on specified topic
  - Asynchronous (not part of transaction commit)
  - But serialization of data can affect commit speed



Neo4j Streams Plugin - Source

Standard Event format

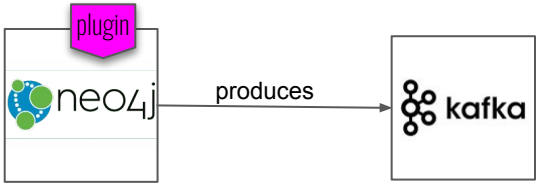
(debezium ‘data change event’ structure)

```
{
  "meta": {
    "timestamp": 1532597182604,
    "username": "neo4j",
    "tx_id": 3,
    "tx_event_id": 0,
    "tx_events_count": 2,
    "operation": "created",        [created/updated/deleted]
    "source": { "hostname": "neo4j.mycompany.com" }
  },
  "payload": {
    "id": "1004",
    "type": "node",                [node/relationship]
    "after": {                     ”before”/”after” sections
      "labels": ["Person"],
      "properties": {
        "email": "xxx@xxx.com",
        "last_name": "Muller",
        "first_name": "Jan"
      }
    }
  },
  "schema": {
    "properties": {...},
    "constraints": [{...}]
  }
 }
```

Neo4j Streams Plugin - Source

Install

`neo4j-streams-4.0.0.jar` file in `/plugins`



Config (in neo4j.conf)

```
kafka.zookeeper.connect=zookeeper:2181
kafka.bootstrap.servers=broker:9093


//3.5
streams.source.topic.nodes.<TOPIC_NAME>=<PATTERN>
streams.source.topic.relationships.<TOPIC_NAME>=<PATTERN>
streams.source.enabled=<true/false, default=true>

//4.0
streams.source.topic.nodes.<TOPIC_NAME>.from.<DB_NAME>=<PATTERN>
streams.source.topic.relationships.<TOPIC_NAME>.from.<DB_NAME>=<PATTERN>
streams.source.enabled.from.<DB_NAME>=<true/false, default=true>
```

Patterns

```
Label{*}

Label1{prop1,prop2};Label3{-prop1}


KNOWS{*};MEETS{prop1, prop2}

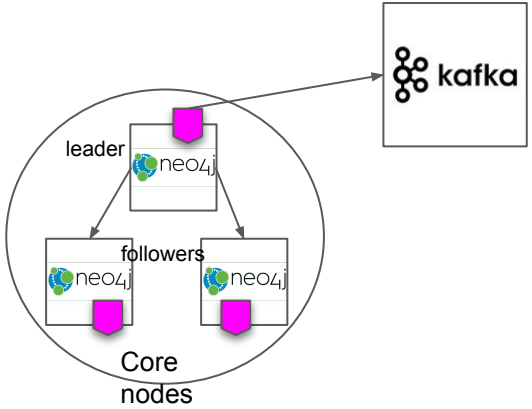
ANSWERS{-prop1,-prop2}
```

Neo4j Streams Plugin - Source

Clusters

- Plugin installed on all core instances
  - leader can change
  - config must be the same on all

- Triggers only on Leader
  - no duplicate events

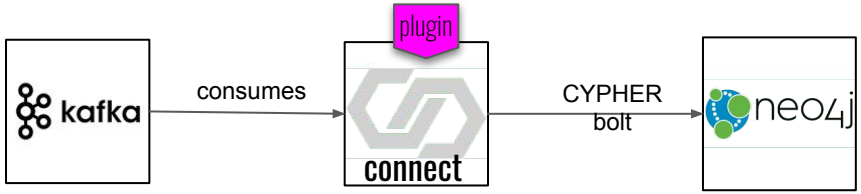




Sink Options

Kafka Connect ‘Neo4j sink’ plugin

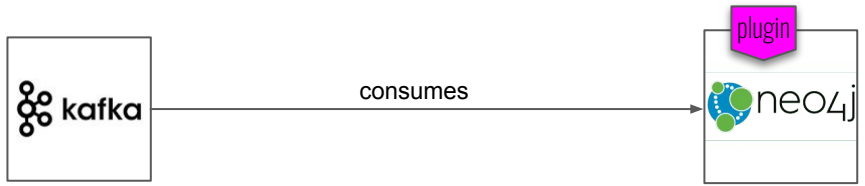
https://www.confluent.io/hub/neo4j/kafka-connect-neo4j



+ Start/stop sink independently from DB
+ Separate hardware requirements & resource consumption
+ Administer with Confluent tools : Connect API, Control Center
+ Can run in parallel

Neo4j Streams plugin - sink

https://neo4j.com/download-center/#integrations



+ Fewer moving parts
+ Enables source, sink & procedures
+ No bolt/network overhead

- Duplicate config/deployment in cluster
- Database restart needed (start/stop procs coming!)



Kafka Connect Plugin

Install

```
confluent-hub install neo4j/kafka-connect-neo4j:1.0.7
```

Config

via `POST /connecthost:8083/connectors`

```
{
  "name": "Neo4jSinkConnector",
  "config": {
    "topics": "my-topic",
    "connector.class": "streams.kafka.connect.sink.Neo4jSinkConnector",
    "neo4j.server.uri": "bolt://neo4j:7687", // bolt+routing for CC
    "neo4j.authentication.basic.username": "neo4j",
    "neo4j.authentication.basic.password": "connect",
    "neo4j.database":"neo4j",
    "neo4j.topic.cypher.my-topic": "MERGE (:Label{name: event.id}"
  }
 }
```

1. consumes events from topic(s)
2. maps data to cypher statements
3. sends cypher to neo4j over bolt connection



CONFLUENT

CONTROLCENTER.CLUST...

CO
Cluster 1

Overview

Brokers

Topics

Connect

Cluster settings

CONNECT CLUSTERS › CONNECT-DEFAULT › CONNECTORS › SINKS ›

Add Connector

01 SETUP CONNECTION | 02 TEST AND VERIFY

Which topics do you want to get data from?

topics: users × stock-trades ×

How should we connect to your data?

Connector class: streams.kafka.connect.sink.Neo4jSinkConnector

name: MyNeo4jConnector

Common

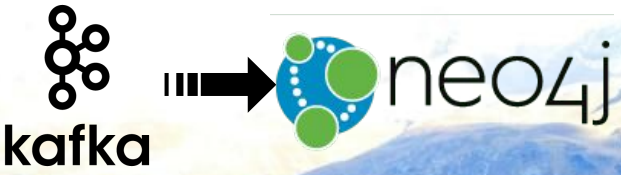
Tasks max: 5

Key converter class: org.apache.kafka.connect.storage.StringConverter

Topics regex:

Value converter class: org.apache.kafka.connect.json.JsonConverter

Which topics do you want to get data from?
How should we connect to your data?
Common
Transforms
Error Handling
Connection
Encryption
Additional Properties



Neo4j Streams Plugin - Sink

Install

`neo4j-streams-4.0.0.jar` file in `/plugins`

Config (in neo4j.conf)

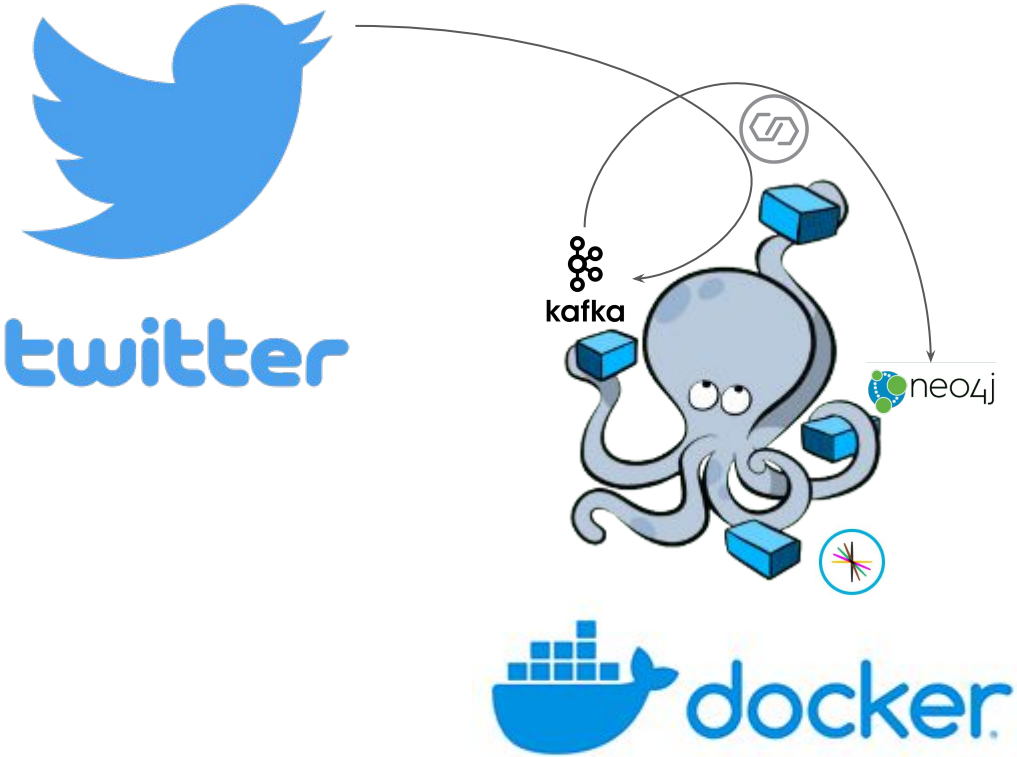
```
kafka.zookeeper.connect=zookeeper:2181
kafka.bootstrap.servers=broker:9093
streams.sink_enabled=true

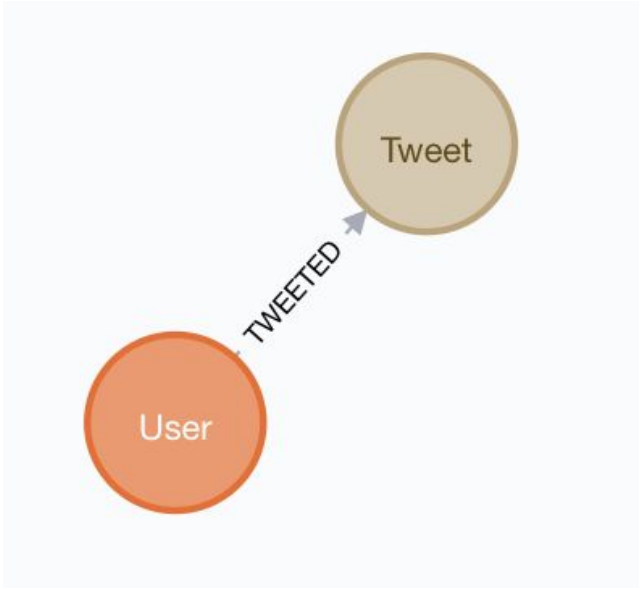
streams.sink.topic.cypher.my-topic.to.<DB_NAME>=MERGE (:Label{name: event.id})
```

1. Connects to kafka
2. Consumes a small batch of events from topic
3. Maps data to cypher statements
4. Run the cypher locally

```
UNWIND [<batch of events>] AS event
MERGE (:Label{name: event.id)
```


kafka
neo4j
Demo


twitter
kafka
neo4j
docker


Tweet
TWEETED
User



Ingestion Strategies

- **Cypher template**
  - most flexible : adapts to any input format, runs any statement
  - “event” variable contains message value

```
streams.sink.topic.cypher.my-topic.to.<DB_NAME>=MERGE (:Label{name: event.id)
```

- **CDC**
  - Debezium format : messages from another neo4j instance or other Debezium sources
  - “sourceId” sub-strategy

```
streams.sink.topic.cdc.sourceId.to.<DB_NAME>my-topic1;my-topic2
```

id (neo4j = internal id) => “sourceId” property
which is used to match nodes
“sourceId” property must be present on existing data

☢ internal id not stable : reused after deletes, restore from backup… ok if source/dest within same cluster

  - “Schema” sub-strategy

```
streams.sink.topic.cdc.schema.to.<DB_NAME>my-topic1;my-topic2
```

Based on unicity constraints

♯ on relationships : rel-key = startNode + endNode + type

Note : use prefix `"streams.sink.*"` in Neo4j plugin / `"neo4j.*"` in Connect

Ingestion Strategies

- **CUD**
  - expects a specific message format
  - custom CDC
    - 1 message = 1 operation on 1 entity in database
      - create/merge/update/delete
  - Requires control of message format

```
streams.sink.topic.cud=my-topic1;my-topic2
```

```
{
  "op": "merge",  [create/merge/update/delete]
  "properties": {
    "foo": "value",
    "key": 1
  },
  "ids": {"key": 1, "otherKey":  "foo"},
  "labels": ["Foo","Bar"],
  "type": "node"       [node/relationship]
 }
```

```
MERGE (n:Foo:Bar {key: event.ids.key,
       otherkey: event.ids.otherkey})
SET n += event.properties
```

- **Node / Relationship pattern**
  - For granular data :
    - 1 message = 1 node or relationship
    - 1 topic per type/label

```
streams.sink.topic.pattern.node.topic-nodes=Person{!userid}
streams.sink.topic.pattern.relationship.topic-rels=(:Person{!userId})-[:KNOWS{since}]->(:Person{!userId2})
```

Note : use prefix *"streams.sink.*"* in Neo4j plugin / *"neo4j.*"* in Connect



Stored Procedures

Config (in neo4j.conf)

```
streams.procedures.enabled.from.<DB_NAME>=<true/false, default=true>
```

```
CALL streams.consume('my-topic', {<config>}) YIELD event
```

```
CALL streams.publish('my-topic', 'Hello World')
```

- can be called by cypher clients
- can be used together with apoc triggers
  - Custom CDC format



Other optional config

Any property prefixed with kafka.* => config for kafka client

```
kafka.auto.offset.reset=earliest
kafka.group.id=neo4j
kafka.enable.auto.commit=true
kafka.key.deserializer=org.apache.kafka.common.serialization.ByteArrayDeserializer
kafka.value.deserializer=org.apache.kafka.common.serialization.ByteArrayDeserializer

kafka.acks=all
kafka.num.partitions=1
kafka.retries=2
kafka.batch.size=16384
kafka.buffer.memory=33554432
```

```
kafka.security.protocol=SSL                //PLAINTEXT, SSL, SASL_PLAINTEXT,
SASL_SSL
kafka.ssl.truststore.location=/path/to/kafka.client.truststore.jks
kafka.ssl.truststore.password=neo4jpassword
kafka.ssl.keystore.location=/path/to/kafka.client.keystore.jks
kafka.ssl.keystore.password=neo4jpassword
kafka.ssl.key.password=neo4jpassword
kafka.ssl.endpoint.identification.algorithm=HTTPS
```

SSL or kerberos authN

Error handling

- No retry for neo4j errors
- DLQ

```
streams.sink.errors.log.enable=true
streams.sink.errors.log.include.messages=true

streams.sink.errors.tolerance=all                    default: none

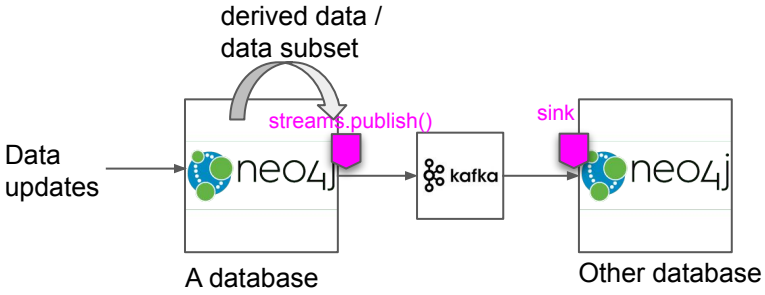
streams.sink.errors.deadletterqueue. topic.name=dlq-topic
streams.sink.errors.deadletterqueue.topic.replication.factor=1
streams.sink.errors.deadletterqueue.context.headers.enable=true
```

Note : use prefix `"streams.sink.*"` in Neo4j plugin / `"neo4j.*"` in Connect

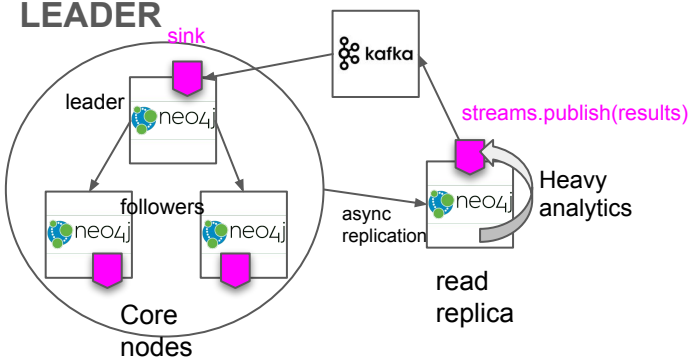


Use Cases

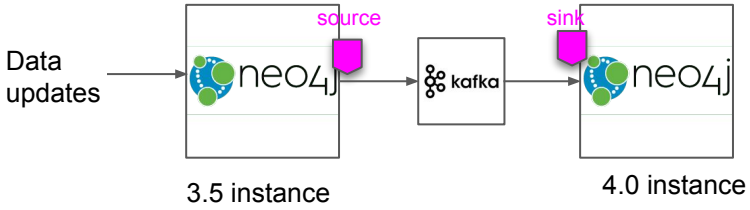
Data pipeline between separate DBMSs with different models



Within cluster : Sync back from RR to LEADER



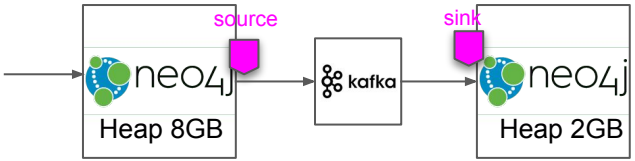
Upgrade/Migration



Performance

https://github.com/neo4j-contrib/neo4j-streams/tree/4.0/performance

- Create node with couple properties
- Local producer/consumer
- Default settings



=> 0.12 ms per node (~7000/s) … but YMMV

Throughput & **latency** depend on :

- Network
- speed of data extraction
  - Pre-process => small, simple messages
  - keep uniform messages on topic
- speed of database transaction
  - simple cypher
  - profile query ; use indexes
- batching parameters (see next slide)



Performance

Batching settings `UNWIND [<batch of events>] AS event`

- `kafka.max.poll.records & neo4j.batch.size`
  - size of batches read from topic / written to Neo4j
  - ideally same value for both
  - start at ~5000-20000, ↗️ until sweet spot.

- `kafka.max.partition.fetch.bytes (default:1MB per partition consumed)`
  - limits memory consumption
  - ↘️ if little heap memory available
  - Ideally : > *batch size * avg record size*

Note : use prefix *"streams.sink.*"* in Neo4j plugin / *"neo4j.*"* in Connect



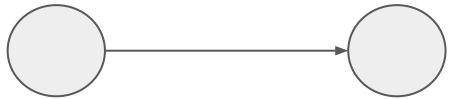
Performance

- Parallelism
  - Use multiple partitions
  - `neo4j.batch.parallelize=true/false` in Connect plugin only

⚠ transactions use **locks**

Can causes waits, potential deadlock errors

Only parallelize with careful design (disjoint graph data)



⚠ CDC

must preserve order of events.