Neo4j Streams: Run with Docker

Introduction

When Neo4j is run in a Docker, some special considerations apply; please see Neo4j Docker Configuration for more information. In particular, the configuration format used in `neo4j.conf` looks different.

Please note that the Neo4j Docker image use a naming convention; you can override every neo4j.conf property by prefix it with `NEO4J_` and using the following transformations:

- single underscore is converted in double underscore: `_ → __`
- point is converted in single underscore: `. → _`

Example:

- `dbms.memory.heap.max_size=8G → NEO4J_dbms_memory_heap_max__size: 8G`
- `dbms.logs.debug.level=DEBUG → NEO4J_dbms_logs_debug_level: DEBUG`

For more information and examples see this section and the Confluent With Docker section of the documentation.

NOTE

Another important thing to watch out for is about possible permissions issue. If you want to running Kafka in Docker using a host volume for which the user is not the owner then you will have a permission error. There are two possible solutions:

- use a root-user
- change permissions of the volume in order to make it accessible by the non-root user

NOTE

The Neo4j docker container is built on an approach that uses environment variables passed to the container as a way to configure Neo4j. There are certain characters which environment variables cannot contain, notably the dash `-` character. Configuring the plugin to use stream names that contain these characters will not work properly, because a configuration environment variable such as `NEO4J_streams_sink_topic_cypher_my-topic` cannot be correctly evaluated as an environment variable (`my-topic`). This is a limitation of the Neo4j docker container rather than neo4j-streams.

Please note that the Neo4j Docker image use a naming convention; you can override every `neo4j.conf` property by prefix it with `NEO4J_` and using the following transformations:

- single underscore is converted in double underscore: `_ → __`
- point is converted in single underscore: `. → _`

Example:

- `dbms.memory.heap.max_size=8G` → `NEO4J_dbms_memory_heap_max__size: 8G`
- `dbms.logs.debug.level=DEBUG` → `NEO4J_dbms_logs_debug_level: DEBUG`

Following you'll find a lightweight Docker Compose file that allows you to test the application in your local environment

Prerequisites:

- Docker
- Docker Compose

Here the instruction about how to configure Docker and Docker-Compose

From the same directory where the compose file is, you can launch this command:

```
docker-compose up -d
```

Source module

Following a compose file that allows you to spin-up Neo4j, Kafka and Zookeeper in order to test the application.

docker-compose.yml

```
version: '3'
services:
  neo4j:
    image: neo4j:3.5
    hostname: neo4j
    container_name: neo4j
    ports:
      - "7474:7474"
      - "7687:7687"
    depends_on:
      - kafka
    volumes:
      - ./neo4j/plugins:/plugins
    environment:
      NEO4J_AUTH: neo4j/streams
      NEO4J_dbms_logs_debug_level: DEBUG
      # KAFKA related configuration
      NEO4J_kafka_zookeeper_connect: zookeeper:12181
      NEO4J_kafka_bootstrap_servers: kafka:19092
      NEO4J_streams_source_topic_nodes_neo4j: Person{*}
      NEO4J_streams_source_topic_relationships_neo4j: KNOWS{*}

  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    hostname: zookeeper
    container_name: zookeeper
    ports:
      - "12181:12181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 12181

  kafka:
    image: confluentinc/cp-kafka:latest
    hostname: kafka
    container_name: kafka
    ports:
      - "19092:19092"
    depends_on:
      - zookeeper
    environment:
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:12181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:19092
```

Launch it locally

Prerequisites

- Install the latest version of Neo4j Streams plugin into `./neo4j/plugins`

Before starting please change the volume directory according to yours, inside the <plugins> dir you must put Streams jar

```
volumes:
    - ./neo4j/plugins:/plugins
```

You can execute a Kafka Consumer that subscribes the topic `neo4j` by executing this command:

```
docker exec kafka kafka-console-consumer --bootstrap-server kafka:19092 --topic neo4j
--from-beginning
```

Then directly from the Neo4j browser you can generate some random data with this query:

```
UNWIND range(1,100) as id
CREATE (p:Person {id:id, name: "Name " + id, age: id % 3}) WITH collect(p) as people
UNWIND people as p1
UNWIND range(1,10) as friend
WITH p1, people[(p1.id + friend) % size(people)] as p2
CREATE (p1)-[:KNOWS {years: abs(p2.id - p1.id)}]->(p2)
```

And if you go back to your consumer you'll see something like this:


```
{"meta":{"timestamp":1571329239766,"username":"neo4j","txId":20,"txEventId":98,"txEven
tsCount":1100,"operation":"created","source":{"hostname":"neo4j"}},"payload":{"id":"84
","before":null,"after":{"properties":{"name":"Name 85","id":85,"age":1},"labels":[
"Person"]},"type":"node"},"schema":{"properties":{"name":"String","id":"Long","age":"L
ong"},"constraints":[]}}

{"meta":{"timestamp":1571329239766,"username":"neo4j","txId":20,"txEventId":99,"txEven
tsCount":1100,"operation":"created","source":{"hostname":"neo4j"}},"payload":{"id":"85
","before":null,"after":{"properties":{"name":"Name 86","id":86,"age":2},"labels":[
"Person"]},"type":"node"},"schema":{"properties":{"name":"String","id":"Long","age":"L
ong"},"constraints":[]}}

{"meta":{"timestamp":1571329239766,"username":"neo4j","txId":20,"txEventId":100,"txEve
ntsCount":1100,"operation":"created","source":{"hostname":"neo4j"}},"payload":{"id":"0
","start":{"id":"0","labels":["Person"],"ids":{}},"end":{"id":"2","labels":["Person"],
"ids":{}},"before":null,"after":{"properties":{"years":2}},"label":"KNOWS","type":"rel
ationship"},"schema":{"properties":{"years":"Long"},"constraints":[]}}
```


Please note that in this example no topic name were specified before the execution of the Kafka Consumer, which is listening on `neo4j` topic. This is because Neo4j Streams plugin, if not specified,

will produce messages into a topic named `neo4j` by default.

Sink module

Following you'll find a simple docker compose file that allow you to spin-up two Neo4j instances one configured as `Source` and one as `Sink`, allowing you to share any data from the `Source` to the `Sink`:

- The `Source` is listening at `http://localhost:8474/browser/` (bolt: `bolt://localhost:8687`)
- The `Sink` is listening at `http://localhost:7474/browser/` (bolt: `bolt://localhost:7687`) and is configured with the `Schema` strategy.

```
    environment:
      NEO4J_streams_sink_enabled: "true"
      NEO4J_streams_sink_topic_neo4j:
        "WITH event.value.payload AS payload, event.value.meta AS meta
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'node' AND meta.operation <>
'deleted' and payload.after.labels[0] = 'Question' THEN [1] ELSE [] END |
          MERGE (n:Question{neo_id: toInteger(payload.id)}) ON CREATE
            SET n += payload.after.properties
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'node' AND meta.operation <>
'deleted' and payload.after.labels[0] = 'Answer' THEN [1] ELSE [] END |
          MERGE (n:Answer{neo_id: toInteger(payload.id)}) ON CREATE
            SET n += payload.after.properties
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'node' AND meta.operation <>
'deleted' and payload.after.labels[0] = 'User' THEN [1] ELSE [] END |
          MERGE (n:User{neo_id: toInteger(payload.id)}) ON CREATE
            SET n += payload.after.properties
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'node' AND meta.operation <>
'deleted' and payload.after.labels[0] = 'Tag' THEN [1] ELSE [] END |
          MERGE (n:Tag{neo_id: toInteger(payload.id)}) ON CREATE
            SET n += payload.after.properties
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'relationship' AND
meta.operation <> 'deleted' and payload.label = 'ANSWERS' THEN [1] ELSE [] END |
          MERGE (s:Answer{neo_id: toInteger(payload.start.id)})
          MERGE (e:Question{neo_id: toInteger(payload.end.id)})
          CREATE (s)-[:ANSWERS{neo_id: toInteger(payload.id)}]->(e)
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'relationship' AND
meta.operation <> 'deleted' and payload.label = 'TAGGED' THEN [1] ELSE [] END |
          MERGE (s:Question{neo_id: toInteger(payload.start.id)})
          MERGE (e:Tag{neo_id: toInteger(payload.end.id)})
          CREATE (s)-[:TAGGED{neo_id: toInteger(payload.id)}]->(e)
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'relationship' AND
meta.operation <> 'deleted' and payload.label = 'PROVIDED' THEN [1] ELSE [] END |
          MERGE (s:User{neo_id: toInteger(payload.start.id)})
          MERGE (e:Answer{neo_id: toInteger(payload.end.id)})
          CREATE (s)-[:PROVIDED{neo_id: toInteger(payload.id)}]->(e)
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'relationship' AND
meta.operation <> 'deleted' and payload.label = 'ASKED' THEN [1] ELSE [] END |
          MERGE (s:User{neo_id: toInteger(payload.start.id)})
          MERGE (e:Question{neo_id: toInteger(payload.end.id)})
          CREATE (s)-[:ASKED{neo_id: toInteger(payload.id)}]->(e)
        )"
```

Launch it locally

In the following example we will use the Neo4j Streams plugin in combination with the APOC procedures (download from here) in order to download some data from Stackoverflow, store them into the Neo4j `Source` instance and replicate these dataset into the `Sink` via the Neo4j Streams plugin.

```
version: '3'

services:
  neo4j-source:
    image: neo4j:3.5
    hostname: neo4j-source
    container_name: neo4j-source
    depends_on:
      - zookeeper
      - broker
    ports:
      - "8474:7474"
      - "8687:7687"
    volumes:
      - ./neo4j/plugins:/plugins
    environment:
      NEO4J_kafka_zookeeper_connect: zookeeper:2181
      NEO4J_kafka_bootstrap_servers: broker:9093
      NEO4J_AUTH: neo4j/source
      NEO4J_dbms_memory_heap_max__size: 2G
      NEO4J_dbms_logs_debug_level: DEBUG
      NEO4J_kafka_batch_size: 16384
      NEO4J_streams_sink_enabled: "false"
      NEO4J_streams_source_schema_polling_interval: 10000

  neo4j-sink:
    image: neo4j:3.5
    hostname: neo4j-sink
    container_name: neo4j-sink
    depends_on:
      - neo4j-source
    ports:
      - "7474:7474"
      - "7687:7687"
    volumes:
      - ./neo4j/plugins-sink:/plugins
    environment:
      NEO4J_kafka_zookeeper_connect: zookeeper:2181
      NEO4J_kafka_bootstrap_servers: broker:9093
      NEO4J_AUTH: neo4j/sink
      NEO4J_dbms_memory_heap_max__size: 2G
      NEO4J_kafka_max_poll_records: 16384
      NEO4J_streams_source_enabled: "false"
```

```
      NEO4J_streams_sink_topic_cdc_schema: "neo4j"
      NEO4J_dbms_logs_debug_level: DEBUG
      NEO4J_streams_sink_enabled: "true"
      NEO4J_streams_sink_topic_neo4j:
        "WITH event.value.payload AS payload, event.value.meta AS meta
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'node' AND meta.operation <>
'deleted' and payload.after.labels[0] = 'Question' THEN [1] ELSE [] END |
          MERGE (n:Question{neo_id: toInteger(payload.id)}) ON CREATE
            SET n += payload.after.properties
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'node' AND meta.operation <>
'deleted' and payload.after.labels[0] = 'Answer' THEN [1] ELSE [] END |
          MERGE (n:Answer{neo_id: toInteger(payload.id)}) ON CREATE
            SET n += payload.after.properties
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'node' AND meta.operation <>
'deleted' and payload.after.labels[0] = 'User' THEN [1] ELSE [] END |
          MERGE (n:User{neo_id: toInteger(payload.id)}) ON CREATE
            SET n += payload.after.properties
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'node' AND meta.operation <>
'deleted' and payload.after.labels[0] = 'Tag' THEN [1] ELSE [] END |
          MERGE (n:Tag{neo_id: toInteger(payload.id)}) ON CREATE
            SET n += payload.after.properties
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'relationship' AND
meta.operation <> 'deleted' and payload.label = 'ANSWERS' THEN [1] ELSE [] END |
          MERGE (s:Answer{neo_id: toInteger(payload.start.id)})
          MERGE (e:Question{neo_id: toInteger(payload.end.id)})
          CREATE (s)-[:ANSWERS{neo_id: toInteger(payload.id)}]->(e)
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'relationship' AND
meta.operation <> 'deleted' and payload.label = 'TAGGED' THEN [1] ELSE [] END |
          MERGE (s:Question{neo_id: toInteger(payload.start.id)})
          MERGE (e:Tag{neo_id: toInteger(payload.end.id)})
          CREATE (s)-[:TAGGED{neo_id: toInteger(payload.id)}]->(e)
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'relationship' AND
meta.operation <> 'deleted' and payload.label = 'PROVIDED' THEN [1] ELSE [] END |
          MERGE (s:User{neo_id: toInteger(payload.start.id)})
          MERGE (e:Answer{neo_id: toInteger(payload.end.id)})
          CREATE (s)-[:PROVIDED{neo_id: toInteger(payload.id)}]->(e)
        )
        FOREACH (ignoreMe IN CASE WHEN payload.type = 'relationship' AND
meta.operation <> 'deleted' and payload.label = 'ASKED' THEN [1] ELSE [] END |
          MERGE (s:User{neo_id: toInteger(payload.start.id)})
          MERGE (e:Question{neo_id: toInteger(payload.end.id)})
          CREATE (s)-[:ASKED{neo_id: toInteger(payload.id)}]->(e)
        )"

  zookeeper:
```

```
    image: confluentinc/cp-zookeeper
    hostname: zookeeper
    container_name: zookeeper
    ports:
      - "2181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000

  broker:
    image: confluentinc/cp-enterprise-kafka
    hostname: broker
    container_name: broker
    depends_on:
      - zookeeper
    ports:
      - "9092:9092"
    expose:
      - "9093"
    environment:
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://broker:9093,OUTSIDE://localhost:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,OUTSIDE:PLAINTEXT
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9093,OUTSIDE://0.0.0.0:9092
      CONFLUENT_METRICS_REPORTER_BOOTSTRAP_SERVERS: broker:9093
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:2181'
      KAFKA_METRIC_REPORTERS: io.confluent.metrics.reporter.ConfluentMetricsReporter
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
      CONFLUENT_METRICS_REPORTER_ZOOKEEPER_CONNECT: zookeeper:2181
      CONFLUENT_METRICS_REPORTER_TOPIC_REPLICAS: 1
      CONFLUENT_METRICS_ENABLE: 'true'
      CONFLUENT_SUPPORT_CUSTOMER_ID: 'anonymous'

  schema_registry:
    image: confluentinc/cp-schema-registry
    hostname: schema_registry
    container_name: schema_registry
    depends_on:
      - zookeeper
      - broker
    ports:
      - "8081:8081"
    environment:
      SCHEMA_REGISTRY_HOST_NAME: schema_registry
      SCHEMA_REGISTRY_KAFKASTORE_CONNECTION_URL: 'zookeeper:2181'
```

Prerequisites

- Install the APOC into `./neo4j/plugins`.
- Install the Neo4j Streams plugin into `./neo4j/plugins` and `./neo4j/plugins-sink`

Import the data

Let's go to two instances in order to create the constraints on both sides:

```
// enable the multi-statement execution:
https://stackoverflow.com/questions/21778435/multiple-unrelated-queries-in-neo4j-
cypher?answertab=votes#tab-top
CREATE CONSTRAINT ON (u:User) ASSERT u.id IS UNIQUE;
CREATE CONSTRAINT ON (a:Answer) ASSERT a.id IS UNIQUE;
CREATE CONSTRAINT ON (t:Tag) ASSERT t.name IS UNIQUE;
CREATE CONSTRAINT ON (q:Question) ASSERT q.id IS UNIQUE;
```

please take a look at the property inside the compose file:

```
NEO4J_streams_source_schema_polling_interval: 10000
```

this means that every 10 seconds the Streams plugin polls the DB in order to retrieve schema changes and store them. So after you created the indexes you need almost to wait 10 seconds before the next step, otherwise the

Now lets go to the `Source` and, in order to import the Stackoverflow dataset, execute the following query:

```
UNWIND range(1, 1) as page
CALL
apoc.load.json("https://api.stackexchange.com/2.2/questions?pagesize=100&order=desc&so
rt=creation&tagged=neo4j&site=stackoverflow&page=" + page) YIELD value
UNWIND value.items AS event
MERGE (question:Question {id:event.question_id}) ON CREATE
  SET question.title = event.title, question.share_link = event.share_link,
question.favorite_count = event.favorite_count

FOREACH (ignoreMe in CASE WHEN exists(event.accepted_answer_id) THEN [1] ELSE [] END |
MERGE (question)<-[:ANSWERS]-(answer:Answer{id: event.accepted_answer_id}))

WITH * WHERE NOT event.owner.user_id IS NULL
MERGE (owner:User {id:event.owner.user_id}) ON CREATE SET owner.display_name =
event.owner.display_name
MERGE (owner)-[:ASKED]->(question)
```

Once the import process has finished to be sure that the data is correctly replicated into the `Sink` execute this query both in `Source` and `Sink` and compare the results:

```
MATCH (n)
RETURN
DISTINCT labels(n),
count(*) AS NumofNodes,
avg(size(keys(n))) AS AvgNumOfPropPerNode,
min(size(keys(n))) AS MinNumPropPerNode,
max(size(keys(n))) AS MaxNumPropPerNode,
avg(size((n)-[]-())) AS AvgNumOfRelationships,
min(size((n)-[]-())) AS MinNumOfRelationships,
max(size((n)-[]-())) AS MaxNumOfRelationships
order by NumofNodes desc
```

You can also launch a Kafka Consumer that subscribes the topic `neo4j` by executing this command:

```
docker exec broker kafka-console-consumer --bootstrap-server broker:9093 --topic neo4j
--from-beginning
```

You'll see something like this:


```
{"meta":{"timestamp":1571403896987,"username":"neo4j","txId":34,"txEventId":330,"txEve
ntsCount":352,"operation":"created","source":{"hostname":"neo4j-source"}},"payload":{
"id":"94","start":{"id":"186","labels":["User"],"ids":{"id":286795}},"end":{"id":"59",
"labels":["Question"],"ids":{"id":58303891}},"before":null,"after":{"properties":{}},"
label":"ASKED","type":"relationship"},"schema":{"properties":{},"constraints":[]}}

{"meta":{"timestamp":1571403896987,"username":"neo4j","txId":34,"txEventId":331,"txEve
ntsCount":352,"operation":"created","source":{"hostname":"neo4j-source"}},"payload":{
"id":"34","start":{"id":"134","labels":["Answer"],"ids":{"id":58180296}},"end":{"id":"
99","labels":["Question"],"ids":{"id":58169215}},"before":null,"after":{"properties":{
}},"label":"ANSWERS","type":"relationship"},"schema":{"properties":{},"constraints":[]
}}
```