Using Neo4j's APIs

Roland Guijt

MVP | CONSULTANT | TRAINER | AUTHOR

@rolandguijt rolandguijt.com

Module Overview

Bolt

A Bolt client

REST service root

REST node and relationship operations

Cypher via REST

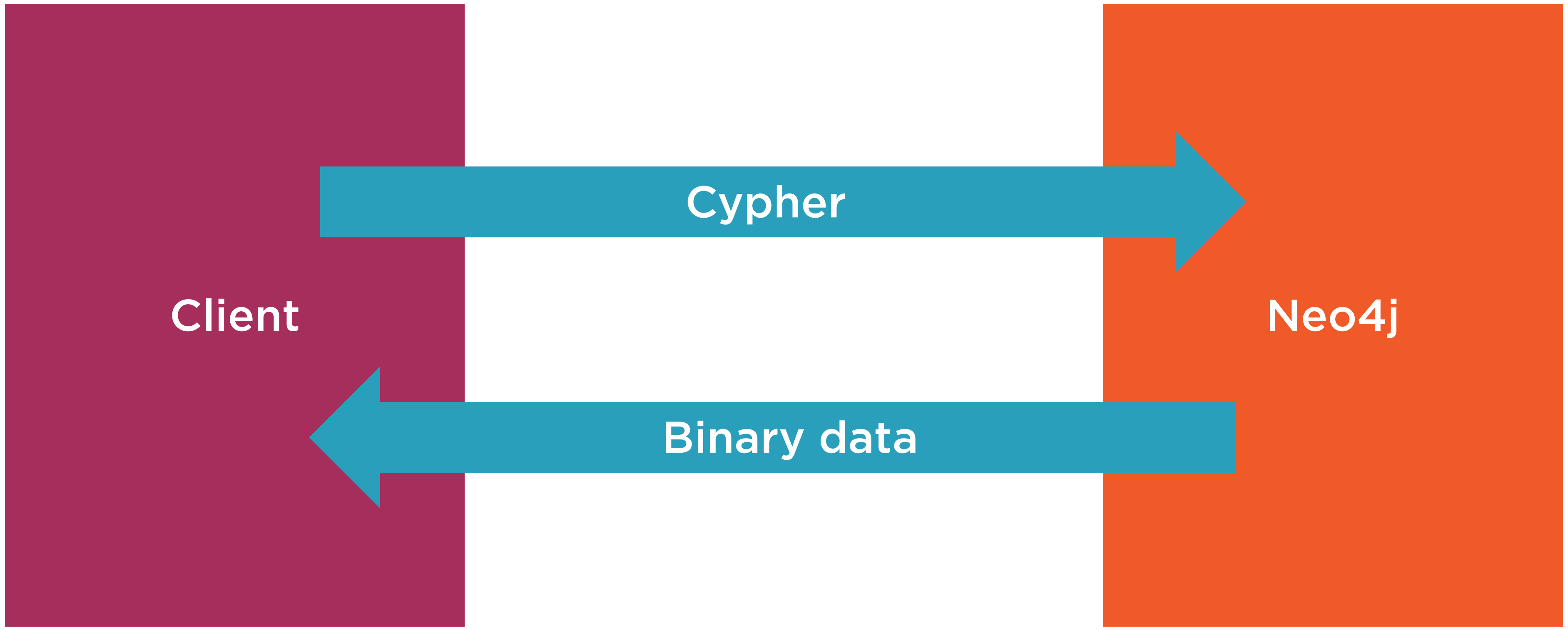
A REST client

Neo4j API Types

REST

Bolt

Bolt



Bolt

Need client library

Performance

No HTTP and uses binary format

Service Root

Provides a REST starting point

Returns list of hypermedia links

GET https://localhost:7474/db/data/

A Typical Request and Response

Request:

```
POST http://someurl
Accept: application/json; charset=UTF-8
Content-Type: application/json
{
        name: “Peter Capaldi”
}
```

Response:

```
201: Created
Content-Length: 1239
Content-Type: application/json; charset=UTF-8
Location: http://localhost:7474/db/data/node/107
{
  <Some Data>
}
```

Service Root

```
GET http://localhost:7474/db/data/
Accept: application/json; charset=UTF-8
```

Node Operations: Get By Id

GET HTTP method

Use node URL returned by service root call

GET http://localhost:7474/db/data/node/1

Node Operations: Create

POST HTTP method

Again use node URL from service root

Returns created node

Node Operations: Delete

DELETE HTTP method

DELETE http://localhost:7474/db/data/node/100

Node Operations: Properties

Use node URL to GET all properties for a node

PUT HTTP method to update/create property on node

Name in URL, value attached

```
PUT
http://localhost:7474/db/data/node/1/properties/salary
Content-Type: application/json
100000
```

Node Operations: Labels

Like properties

GET lists, POST adds, PUT replaces

```
POST http://localhost:7474/db/data/node/1/labels
Content-Type: application/json
[“Person”, “Actor”]
```

Relationship Operations

Like nodes

Use relationship URL

Notable exceptions follow

Relationship Operations: Get for Node

GET http://localhost:7474/db/data/node/1/relationships/all

GET
http://localhost:7474/db/data/node/1/relationships/all
/PLAYED&REGENERATED_TO

Relationship Operations: Create

```
POST
http://localhost:7474/db/data/node/1/relationships
Content-Type: application/json
{
  "to" : "http://localhost:7474/db/data/node/19",
  "type" : "LOVES",
  "data" : {
    “intensity" : “medium"
  }
}
```

Traversals

Traverse the graph

One node as starting point

Paged traversals are stored for later retrieval

URL of starting node

What to return as URL extension path, fullpath, node, relationship

Further details in attachment

Traversals

```
POST http://localhost:7474/db/data/node/1/traverse/node
Content-Type: application/json
{
 "order" : "breadth_first",
 "return_filter" : {
   "body" : "position.endNode().getProperty('name').toLowerCase().contains('p')",
   "language" : "javascript"
 },
 "prune_evaluator" : {
   "body" : "position.length() > 10",
   "language" : "javascript"
 },
 "uniqueness" : "node_global",
 "relationships" : [ {
   "direction" : "out",
   "type" : "REGENERATED_TO"
 }, {
   "direction" : "all",
   "type" : "PLAYED"
 } ],
 "max_depth" : 3
}
```

Traversals

Start

1. Related

2. Related of related

3. Related of related of related

Batch Operations

```
POST http://localhost:7474/db/data/batch
Content-Type: application/json
[ {
  "method" : "POST",
  "to" : "/node",
  "id" : 0,
  "body" : {
    "name" : "bob"
  }
}, {
  "method" : "POST",
  "to" : "/node",
  "id" : 1,
  "body" : {
    "age" : 12
  }
},
 {
  "method" : "POST",
  "to" : "{0}/relationships",
  "id" : 3,
  "body" : {
    "to" : "{1}",
    "data" : {
      "since" : "2010"
    },
    "type" : "KNOWS"
  }
 }
]
```

Transactional Cypher Endpoint

Execute Cypher via the REST API

JSON

Transactional

Transaction can remain open between requests

Transaction can timeout

Begin a Transaction and Commit in One Go

```
POST http://localhost:7474/db/data/transaction/commit
Content-Type: application/json
{
 "statements" : [ {
   "statement" : "CREATE (n {props}) RETURN n",
   "parameters" : {
     "props" : {
       "name" : “Peter Capaldi“,
       “salary” : 100000
     }
   }
 } ]
}
```

Start a Transaction

```
POST http://localhost:7474/db/data/transaction

201: Created
Content-Type: application/json
Location: http://localhost:7474/db/data/transaction/7
{
 "commit" :
"http://localhost:7474/db/data/transaction/7/commit",
 "results" : .....,
 "transaction" : {
   "expires" : "Mon, 2 Feb 2015 20:53:51 +0000"
 }}
```

Execute Subsequent Request In Transaction

POST http://localhost:7474/db/data/transaction/7

POST
http://localhost:7474/db/data/transaction/7/Commit

DELETE http://localhost:7474/db/data/transaction/7

Summary

The Bolt API provides fast access to Neo4j for many platforms

It can execute Cypher statements

A REST style API is also available with many options