

Full Stack Development With

GraphQL + Neo4j



William Lyon
@lyonwj
lyonwj.com



GraphConnect
Powered by Neo4j



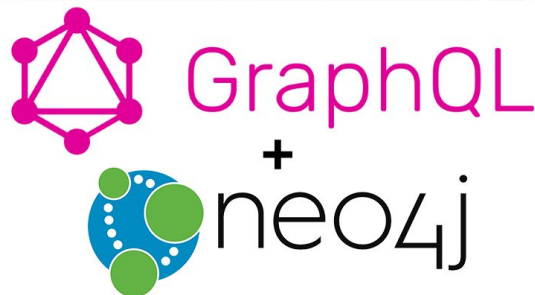
William Lyon

DevRel Engineering @neo4j

will@neo4j.com
[@lyonwj](https://twitter.com/lyonwj)
lyonwj.com

GraphQL + Neo4j

- An overview of GraphQL
- Building a GraphQL service
- Neo4j-GraphQL integration(s)



An Overview Of GraphQL

- A new paradigm for building APIs
- Schema definition
 - Types
 - GraphQL entry points (Query & Mutation types)
- Query language for APIs
 - Limited support for “queries” (aggregations, filtering, ...)
- Community of tools
 - GraphiQL
 - Mocking
 - Performance monitoring



GraphQL

GraphiQL



Prettify

History

< Query

Movie



Q Search Movie...

No Description

FIELDS

moviend: String!

title: String

year: Int

plot: String

poster: String

imdbRating: Float

genres: [String]

similar: [Movie]

```
1 query MovieSearch($title: String!, $limit: Int!) {
2   {
3     movies(subString:$title, limit:$limit) {
4       movieId
5       title
6       year
7       genres
8       poster
9       plot
10      imdbRating
11      similar {
12        title
13        poster
14        year
15      }
16    }
17  }
18 }
```

QUERY VARIABLES

```
1 {
2   "title": "River Runs Through It",
3   "limit": 3
4 }
```

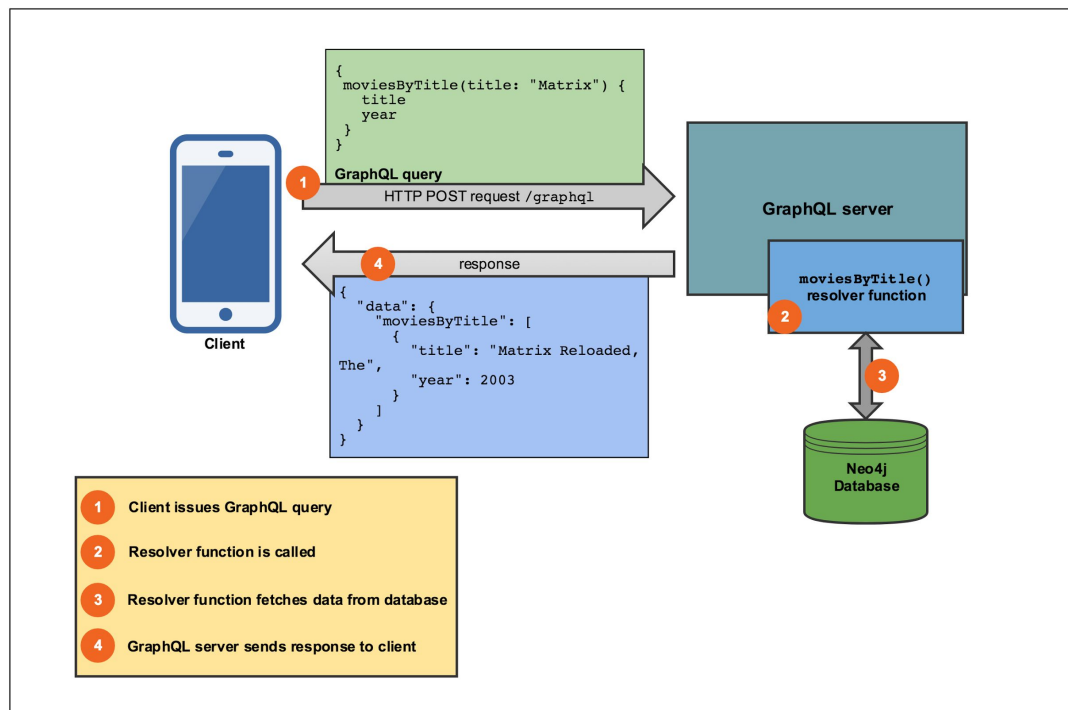
```
{
  "data": {
    "movies": [
      {
        "movieId": "3100",
        "title": "River Runs Through It, A",
        "year": 1992,
        "genres": [
          "Drama"
        ],
        "poster": "http://ia.media-
imdb.com/images/M/MV5BMTM2Nzc5MjI4NF5BML5BanBnXkFtZTYwNzgWjc5._V1_SX300.jpg",
        "plot": "The story about two sons of a stern minister -- one reserved,
one rebellious -- growing up in rural Montana while devoted to fly fishing.",
        "imdbRating": 7.3,
        "similar": [
          {
            "title": "Forrest Gump",
            "poster": "http://ia.media-
imdb.com/images/M/MV5BMTI1Nzk1MzQwMV5BML5BanBnXkFtZTYwODkxOTA5._V1_SX300.jpg",
            "year": 1994
          },
          {
            "title": "Titanic",
            "poster": "http://ia.media-
imdb.com/images/M/MV5BMjExNzQwNDM0N15BML5BanBnXkFtZTcwMzkxOTUwNw@._V1_SX300.jp
g",
            "year": 1997
          },
          {
            "title": "Shawshank Redemption, The",
            "poster": "http://ia.media-
imdb.com/images/M/MV5BODU4MjU4NjIwN15BML5BanBnXkFtZTgwMDU2MjEyMDE@._V1_SX300.jp
g",
            "year": 1994
          }
        ]
      }
    ]
  }
}
```

[bit.ly/ graphiq1](https://bit.ly/graphiq1)

Building A GraphQL Service

- 1) Define a schema
- 2) Implement resolver functions
 - Fetch data from data layer

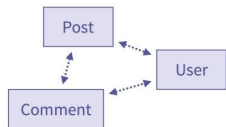
Building A GraphQL Service



<https://dzone.com/refcardz/an-overview-of-graphql>

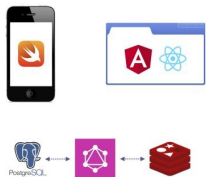
GraphQL First Development

GraphQL First Development



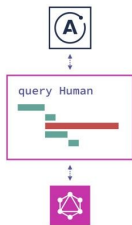
1. Design API schema

Contract between frontend and backend with a shared schema language



2. Build UI and backend

Parallelize with mocking, develop component-based UIs with GraphQL containers



3. Run in production

Static queries make loading predictable, schema tells you which fields are being used

1. Design API schema
2. Build UI and backend
3. Deploy!

- Schema is your friend
- GraphQL Schema is the API spec
 - Allows for simultaneous frontend and backend development
 - Enables introspection
 - Build other tools (graphql)

IDL Schema Syntax



```
type Movie {  
  movieId: ID!  
  title: String  
  year: Int  
  plot: String  
  poster: String  
  imdbRating: Float  
  genres: [String]  
  similar(first: Int=3, offset:Int=0): [Movie]  
}
```

```
type Query {  
  moviesByTitle(subString: String!, first: Int=3, offset: Int=0): [Movie]  
}
```



GraphQL Resolver Functions



```
Query: {  
  moviesByTitle: (root, args, context) => {  
    let session = context.driver.session();  
    let query = "MATCH (movie:Movie) WHERE movie.title CONTAINS $subString RETURN movie LIMIT $first;";  
    return session.run(query, args)  
      .then( result => { return result.records.map(record => { return record.get("movie").properties })})  
  },  
}
```

GraphQL Resolver Functions

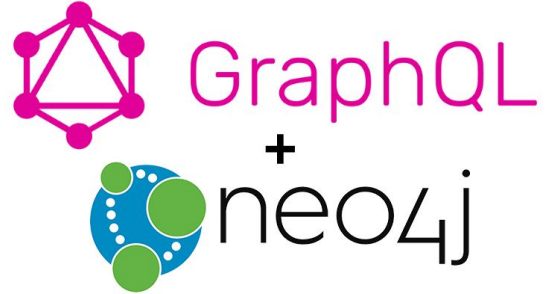


```
const resolvers = {
  Query: {
    moviesByTitle: (root, args, context) => {
      let session = context.driver.session();
      let query = "MATCH (movie:Movie) WHERE movie.title CONTAINS $subString RETURN movie LIMIT $first;";
      return session.run(query, args)
        .then( result => { return result.records.map(record => { return record.get("movie").properties });})
    },
  },
  Movie: {
    genres: (movie, _, context) => {
      let session = context.driver.session();
      let params = {movieId: movie.movieId};
      let query = `
        MATCH(m:Movie)-[:IN_GENRE]->(g:Genre)
        WHERE m.movieId = $movieId
        RETURN g.name AS genre
      `;
      return session.run(query, params)
        .then( result => { return result.records.map(record => {return record.get("genre")});})
    },
    similar: (movie, _, context) => {
      let session = context.driver.session();
      let params = {movieId: movie.movieId};
      let query = `
        MATCH (m:Movie) WHERE m.movieId = $movieId
        MATCH (m)-[:IN_GENRE]->(g:Genre)<-[:IN_GENRE]-(movie:Movie)
        WITH m, movie, COUNT(*) AS genreOverlap
        MATCH (m)<-[:RATED]-(User)-[:RATED]->(movie:Movie)
        WITH movie,genreOverlap, COUNT(*) AS userRatedScore
        RETURN movie ORDER BY (0.9 * genreOverlap) + (0.1 * userRatedScore)  DESC LIMIT 3
      `;
      return session.run(query, params)
        .then( result => {return result.records.map(record => {return record.get("movie").properties});})
    }
  }
};
```

<https://launchpad.graphql.com/x57134gwl>

A GraphQL - Neo4j Integration?

- Developer productivity
- Translate GraphQL → Cypher?
- Improve performance?
- Expose Cypher through GraphQL?



Neo4j-GraphQL Extension

Note This branch is for supporting Neo4j 3.2.

build passing

This implementation provides a GraphQL API to Neo4j, it comes as library but can also be installed as Neo4j server extension to act as a GraphQL endpoint. It turns GraphQL queries and mutations into Cypher statements and executes them on Neo4j.

We want to explore three approaches:

1. read schema / metadata from the database provide GraphQL DataFetcher that generate and run Cypher (WIP) ✓
2. make the same work with externally configured schema information (using IDL) ✓

github.com/neo4j-graphql/neo4j-graphql

circleci passing

neo4j-graphql-js

A GraphQL to Cypher query execution layer for Neo4j and JavaScript GraphQL implementations.

neo4j-graphql-js is in early development. There are rough edges and APIs may change. Please file issues for any bugs that you find or feature requests.

Installation and usage

Install

```
npm install --save neo4j-graphql-js
```

Then call `neo4jgraphql()` in your GraphQL resolver. Your GraphQL query will be translated to Cypher and the query passed to Neo4j.

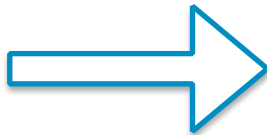
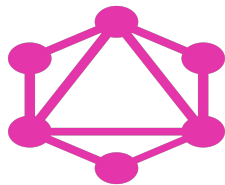
```
import {neo4jgraphql} from 'neo4j-graphql-js';

const resolvers = {
  Query: {
    Movie(object, params, ctx, resolveInfo) {
      return neo4jgraphql(object, params, ctx, resolveInfo);
    }
  }
};
```

github.com/neo4j-graphql/neo4j-graphql-js

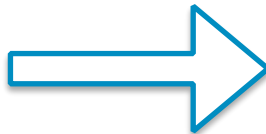
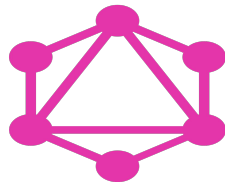
Use The Schema, Luke

```
1  type Movie {
2      movieId: ID!
3      title: String
4      year: Int
5      plot: String
6      poster: String
7      imdbRating: Float
8      genres: [String]
9      similar(first: Int = 3, offset: Int = 0): [Movie] @cypher(statement: "WITH {this} AS this MATCH (this)--(:Genre)--(o:Movie) RETURN o")
10     mostSimilar: Movie @cypher(statement: "WITH {this} AS this RETURN this")
11     degree: Int @cypher(statement: "WITH {this} AS this RETURN SIZE((this)--())")
12     actors(first: Int = 3, offset: Int = 0): [Actor] @relation(name: "ACTED_IN", direction: "IN")
13     avgStars: Float
14     filmedIn: State @relation(name: "FILMED_IN", direction: "OUT")
15 }
16
17 type State {
18     name: String
19 }
20
21 type Actor {
22     id: ID!
23     name: String
24     movies: [Movie]
25 }
26
27 type User {
28     id: ID!
29     name: String
30 }
31
32 type Query {
33     Movie(id: ID, title: String, year: Int, plot: String, poster: String, imdbRating: Float, first: Int, offset: Int): [Movie]
34     MoviesByYear(year: Int): [Movie]
35     AllMovies: [Movie]
36     MovieById(movieId: ID!): Movie
37 }
```



openCypher

Use The Schema, Luke



openCypher

```
{  
  Movie(title: "River Runs Through It, A") {  
    title  
    year  
    imdbRating  
  }  
}
```

```
MATCH (movie:Movie {title:"River Runs Through It, A"})  
RETURN movie { .title , .year , .imdbRating } AS movie  
SKIP 0
```

<https://github.com/neo4j-graphql/>

Improved Performance

```
{  
  Movie(title: "River Runs Through It, A") {  
    title  
    year  
    imdbRating  
    actors {  
      name  
    }  
  }  
}
```



```
MATCH (movie:Movie {title:"River Runs Through It, A"})  
RETURN movie { .title , .year , .imdbRating,  
  actors: [(movie)-[ACTED_IN]-(movie_actors:Actor) | movie_actors { .name }] }  
AS movie  
SKIP 0
```

openCypher

- N+1 query problem
 - Batching
- GraphQL → single Cypher query
 - Single round trip to database

Expose Cypher in GraphQL

- GraphQL directives
- **@cypher** schema directive
 - Map GraphQL fields to a Cypher query

```
type Movie {  
  movieId: ID!  
  title: String  
  year: Int  
  plot: String  
  poster: String  
  imdbRating: Float  
  genres: [String]  
  similar(first: Int = 3, offset: Int = 0): [Movie] @cypher(statement: "MATCH (this)--(:Genre)--(o:Movie) RETURN o")  
  degree: Int @cypher(statement: "RETURN SIZE((this)--())")  
  actors(first: Int = 3, offset: Int = 0): [Actor] @relation(name: "ACTED_IN", direction: "IN")  
  avgStars: Float  
  filmedIn: State @relation(name: "FILMED_IN", direction: "OUT")  
}
```


@cypher Schema Directives

```
{  
  Movie(title: "River Runs Through It, A") {  
    title  
    year  
    imdbRating  
    actors {  
      name  
    }  
    similar(first: 3) {  
      title  
    }  
  }  
}
```



```
MATCH (movie:Movie {title:"River Runs Through It, A"})  
RETURN movie { .title , .year , .imdbRating,  
  actors: [(movie)->[ACTED_IN]-(movie_actors:Actor) | movie_actors { .name }],  
  similar: [ x IN apoc.cypher.runFirstColumn("  
    WITH {this} AS this  
    MATCH (this)-[:IN_GENRE]->(:Genre)<-[:IN_GENRE]-(o:Movie)  
    RETURN o",  
    {this: movie}, true) | x { .title }][..3]  
  ] } AS movie  
SKIP 0
```

openCypher

- Still a single Cypher query / single round-trip
 - **@cypher** annotated query becomes a sub-query

Auto-generate Cypher queries

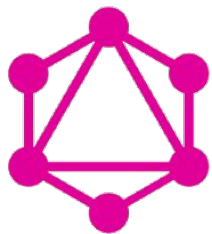
Works with apollo-server, graphql-tools, graphql-js,...

The image shows a VS Code editor with a GraphQL schema and query. The schema defines types for Movie, Person, and User, and includes resolvers and a query. The query is a GraphQL query that fetches movies by title substring.

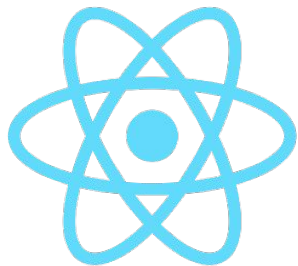
```

1 // Welcome to Launchpad!
2 // Log in to edit and save pads, run queries in GraphQL on the right.
3 // Click "Download" above to get a zip with a standalone Node.js server.
4 // See docs and examples at: https://github.com/apollolibrary/awesome-graphql
5 // GraphQL-tools combines a schema string with resolvers.
6 import { makeExecutableSchema } from 'graphql-tools';
7 import { vi as neo4j } from 'neo4j-driver';
8 import { neo4jgraphql } from 'neo4j-graphql-js';
9
10 const typeDefs = `
11 type Movie {
12   movieId: ID!
13   title: String!
14   year: Int!
15   plot: String!
16   poster: String!
17   imdbRating: Float!
18   genres: String!
19   similar(first: Int = 3, offset: Int = 0): [Movie] @cypher(statement: "MATCH (this)--(:Genre)--(aMovie) RETURN mostSimilar: Movie @cypher(statement: "WITH (this) AS this RETURN this")
20 degree: Int @cypher(statement: "WITH (this) AS this RETURN SIZE((this)-()-())")
21 actors(first: Int = 3, offset: Int = 0): [Actor] @relation(name: "ACTED_IN", direction: "IN")
22 avgStars: Float!
23 }
24
25 interface Person {
26   id: ID!
27   name: String!
28 }
29
30 type Actor implements Person {
31   id: ID!
32   name: String!
33   movies: [Movie]!
34 }
35
36 type User implements Person {
37   id: ID!
38   name: String!
39 }
40
41 type Query {
42   movie(id: ID!, title: String!, year: Int!, plot: String!, poster: String!, imdbRating: Float!, first: Int!, offset: Int!): Movie!
43 }
44
45 const resolvers = {
46   // root entry point to GraphQL service
47   Query: {
48     // fetch movies by title substring
49     movies: (obj, params, ctx, resolvers) => {
50       // neo4jgraphql respects the GraphQL query and schema to generate a single Cypher query
51     }
52   }
53 }
54
55 const schema = makeExecutableSchema({ typeDefs, resolvers });
56
57 const driver = neo4j.driver('neo4j://localhost:7687', neo4j.auth.basic('neo4j', 'password'));
58
59 const neo4jgraphqlPlugin = neo4jgraphql({ driver, schema });
60
61 const { neo4jgraphql } = neo4jgraphqlPlugin;
62
63 const { neo4jgraphql } = neo4jgraphqlPlugin;
64
65 const { neo4jgraphql } = neo4jgraphqlPlugin;
66
67 const { neo4jgraphql } = neo4jgraphqlPlugin;
68
69 const { neo4jgraphql } = neo4jgraphqlPlugin;
70
71 const { neo4jgraphql } = neo4jgraphqlPlugin;
72
73 const { neo4jgraphql } = neo4jgraphqlPlugin;
74
75 const { neo4jgraphql } = neo4jgraphqlPlugin;
76
77 const { neo4jgraphql } = neo4jgraphqlPlugin;
78
79 const { neo4jgraphql } = neo4jgraphqlPlugin;
80
81 const { neo4jgraphql } = neo4jgraphqlPlugin;
82
83 const { neo4jgraphql } = neo4jgraphqlPlugin;
84
85 const { neo4jgraphql } = neo4jgraphqlPlugin;
86
87 const { neo4jgraphql } = neo4jgraphqlPlugin;
88
89 const { neo4jgraphql } = neo4jgraphqlPlugin;
90
91 const { neo4jgraphql } = neo4jgraphqlPlugin;
92
93 const { neo4jgraphql } = neo4jgraphqlPlugin;
94
95 const { neo4jgraphql } = neo4jgraphqlPlugin;
96
97 const { neo4jgraphql } = neo4jgraphqlPlugin;
98
99 const { neo4jgraphql } = neo4jgraphqlPlugin;
100
101 const { neo4jgraphql } = neo4jgraphqlPlugin;
102
103 const { neo4jgraphql } = neo4jgraphqlPlugin;
104
105 const { neo4jgraphql } = neo4jgraphqlPlugin;
106
107 const { neo4jgraphql } = neo4jgraphqlPlugin;
108
109 const { neo4jgraphql } = neo4jgraphqlPlugin;
110
111 const { neo4jgraphql } = neo4jgraphqlPlugin;
112
113 const { neo4jgraphql } = neo4jgraphqlPlugin;
114
115 const { neo4jgraphql } = neo4jgraphqlPlugin;
116
117 const { neo4jgraphql } = neo4jgraphqlPlugin;
118
119 const { neo4jgraphql } = neo4jgraphqlPlugin;
120
121 const { neo4jgraphql } = neo4jgraphqlPlugin;
122
123 const { neo4jgraphql } = neo4jgraphqlPlugin;
124
125 const { neo4jgraphql } = neo4jgraphqlPlugin;
126
127 const { neo4jgraphql } = neo4jgraphqlPlugin;
128
129 const { neo4jgraphql } = neo4jgraphqlPlugin;
130
131 const { neo4jgraphql } = neo4jgraphqlPlugin;
132
133 const { neo4jgraphql } = neo4jgraphqlPlugin;
134
135 const { neo4jgraphql } = neo4jgraphqlPlugin;
136
137 const { neo4jgraphql } = neo4jgraphqlPlugin;
138
139 const { neo4jgraphql } = neo4jgraphqlPlugin;
140
141 const { neo4jgraphql } = neo4jgraphqlPlugin;
142
143 const { neo4jgraphql } = neo4jgraphqlPlugin;
144
145 const { neo4jgraphql } = neo4jgraphqlPlugin;
146
147 const { neo4jgraphql } = neo4jgraphqlPlugin;
148
149 const { neo4jgraphql } = neo4jgraphqlPlugin;
150
151 const { neo4jgraphql } = neo4jgraphqlPlugin;
152
153 const { neo4jgraphql } = neo4jgraphqlPlugin;
154
155 const { neo4jgraphql } = neo4jgraphqlPlugin;
156
157 const { neo4jgraphql } = neo4jgraphqlPlugin;
158
159 const { neo4jgraphql } = neo4jgraphqlPlugin;
160
161 const { neo4jgraphql } = neo4jgraphqlPlugin;
162
163 const { neo4jgraphql } = neo4jgraphqlPlugin;
164
165 const { neo4jgraphql } = neo4jgraphqlPlugin;
166
167 const { neo4jgraphql } = neo4jgraphqlPlugin;
168
169 const { neo4jgraphql } = neo4jgraphqlPlugin;
170
171 const { neo4jgraphql } = neo4jgraphqlPlugin;
172
173 const { neo4jgraphql } = neo4jgraphqlPlugin;
174
175 const { neo4jgraphql } = neo4jgraphqlPlugin;
176
177 const { neo4jgraphql } = neo4jgraphqlPlugin;
178
179 const { neo4jgraphql } = neo4jgraphqlPlugin;
180
181 const { neo4jgraphql } = neo4jgraphqlPlugin;
182
183 const { neo4jgraphql } = neo4jgraphqlPlugin;
184
185 const { neo4jgraphql } = neo4jgraphqlPlugin;
186
187 const { neo4jgraphql } = neo4jgraphqlPlugin;
188
189 const { neo4jgraphql } = neo4jgraphqlPlugin;
190
191 const { neo4jgraphql } = neo4jgraphqlPlugin;
192
193 const { neo4jgraphql } = neo4jgraphqlPlugin;
194
195 const { neo4jgraphql } = neo4jgraphqlPlugin;
196
197 const { neo4jgraphql } = neo4jgraphqlPlugin;
198
199 const { neo4jgraphql } = neo4jgraphqlPlugin;
200
201 const { neo4jgraphql } = neo4jgraphqlPlugin;
202
203 const { neo4jgraphql } = neo4jgraphqlPlugin;
204
205 const { neo4jgraphql } = neo4jgraphqlPlugin;
206
207 const { neo4jgraphql } = neo4jgraphqlPlugin;
208
209 const { neo4jgraphql } = neo4jgraphqlPlugin;
210
211 const { neo4jgraphql } = neo4jgraphqlPlugin;
212
213 const { neo4jgraphql } = neo4jgraphqlPlugin;
214
215 const { neo4jgraphql } = neo4jgraphqlPlugin;
216
217 const { neo4jgraphql } = neo4jgraphqlPlugin;
218
219 const { neo4jgraphql } = neo4jgraphqlPlugin;
220
221 const { neo4jgraphql } = neo4jgraphqlPlugin;
222
223 const { neo4jgraphql } = neo4jgraphqlPlugin;
224
225 const { neo4jgraphql } = neo4jgraphqlPlugin;
226
227 const { neo4jgraphql } = neo4jgraphqlPlugin;
228
229 const { neo4jgraphql } = neo4jgraphqlPlugin;
230
231 const { neo4jgraphql } = neo4jgraphqlPlugin;
232
233 const { neo4jgraphql } = neo4jgraphqlPlugin;
234
235 const { neo4jgraphql } = neo4jgraphqlPlugin;
236
237 const { neo4jgraphql } = neo4jgraphqlPlugin;
238
239 const { neo4jgraphql } = neo4jgraphqlPlugin;
240
241 const { neo4jgraphql } = neo4jgraphqlPlugin;
242
243 const { neo4jgraphql } = neo4jgraphqlPlugin;
244
245 const { neo4jgraphql } = neo4jgraphqlPlugin;
246
247 const { neo4jgraphql } = neo4jgraphqlPlugin;
248
249 const { neo4jgraphql } = neo4jgraphqlPlugin;
250
251 const { neo4jgraphql } = neo4jgraphqlPlugin;
252
253 const { neo4jgraphql } = neo4jgraphqlPlugin;
254
255 const { neo4jgraphql } = neo4jgraphqlPlugin;
256
257 const { neo4jgraphql } = neo4jgraphqlPlugin;
258
259 const { neo4jgraphql } = neo4jgraphqlPlugin;
260
261 const { neo4jgraphql } = neo4jgraphqlPlugin;
262
263 const { neo4jgraphql } = neo4jgraphqlPlugin;
264
265 const { neo4jgraphql } = neo4jgraphqlPlugin;
266
267 const { neo4jgraphql } = neo4jgraphqlPlugin;
268
269 const { neo4jgraphql } = neo4jgraphqlPlugin;
270
271 const { neo4jgraphql } = neo4jgraphqlPlugin;
272
273 const { neo4jgraphql } = neo4jgraphqlPlugin;
274
275 const { neo4jgraphql } = neo4jgraphqlPlugin;
276
277 const { neo4jgraphql } = neo4jgraphqlPlugin;
278
279 const { neo4jgraphql } = neo4jgraphqlPlugin;
280
281 const { neo4jgraphql } = neo4jgraphqlPlugin;
282
283 const { neo4jgraphql } = neo4jgraphqlPlugin;
284
285 const { neo4jgraphql } = neo4jgraphqlPlugin;
286
287 const { neo4jgraphql } = neo4jgraphqlPlugin;
288
289 const { neo4jgraphql } = neo4jgraphqlPlugin;
290
291 const { neo4jgraphql } = neo4jgraphqlPlugin;
292
293 const { neo4jgraphql } = neo4jgraphqlPlugin;
294
295 const { neo4jgraphql } = neo4jgraphqlPlugin;
296
297 const { neo4jgraphql } = neo4jgraphqlPlugin;
298
299 const { neo4jgraphql } = neo4jgraphqlPlugin;
300
301 const { neo4jgraphql } = neo4jgraphqlPlugin;
302
303 const { neo4jgraphql } = neo4jgraphqlPlugin;
304
305 const { neo4jgraphql } = neo4jgraphqlPlugin;
306
307 const { neo4jgraphql } = neo4jgraphqlPlugin;
308
309 const { neo4jgraphql } = neo4jgraphqlPlugin;
310
311 const { neo4jgraphql } = neo4jgraphqlPlugin;
312
313 const { neo4jgraphql } = neo4jgraphqlPlugin;
314
315 const { neo4jgraphql } = neo4jgraphqlPlugin;
316
317 const { neo4jgraphql } = neo4jgraphqlPlugin;
318
319 const { neo4jgraphql } = neo4jgraphqlPlugin;
320
321 const { neo4jgraphql } = neo4jgraphqlPlugin;
322
323 const { neo4jgraphql } = neo4jgraphqlPlugin;
324
325 const { neo4jgraphql } = neo4jgraphqlPlugin;
326
327 const { neo4jgraphql } = neo4jgraphqlPlugin;
328
329 const { neo4jgraphql } = neo4jgraphqlPlugin;
330
331 const { neo4jgraphql } = neo4jgraphqlPlugin;
332
333 const { neo4jgraphql } = neo4jgraphqlPlugin;
334
335 const { neo4jgraphql } = neo4jgraphqlPlugin;
336
337 const { neo4jgraphql } = neo4jgraphqlPlugin;
338
339 const { neo4jgraphql } = neo4jgraphqlPlugin;
340
341 const { neo4jgraphql } = neo4jgraphqlPlugin;
342
343 const { neo4jgraphql } = neo4jgraphqlPlugin;
344
345 const { neo4jgraphql } = neo4jgraphqlPlugin;
346
347 const { neo4jgraphql } = neo4jgraphqlPlugin;
348
349 const { neo4jgraphql } = neo4jgraphqlPlugin;
350
351 const { neo4jgraphql } = neo4jgraphqlPlugin;
352
353 const { neo4jgraphql } = neo4jgraphqlPlugin;
354
355 const { neo4jgraphql } = neo4jgraphqlPlugin;
356
357 const { neo4jgraphql } = neo4jgraphqlPlugin;
358
359 const { neo4jgraphql } = neo4jgraphqlPlugin;
360
361 const { neo4jgraphql } = neo4jgraphqlPlugin;
36
```

The GRAND stack



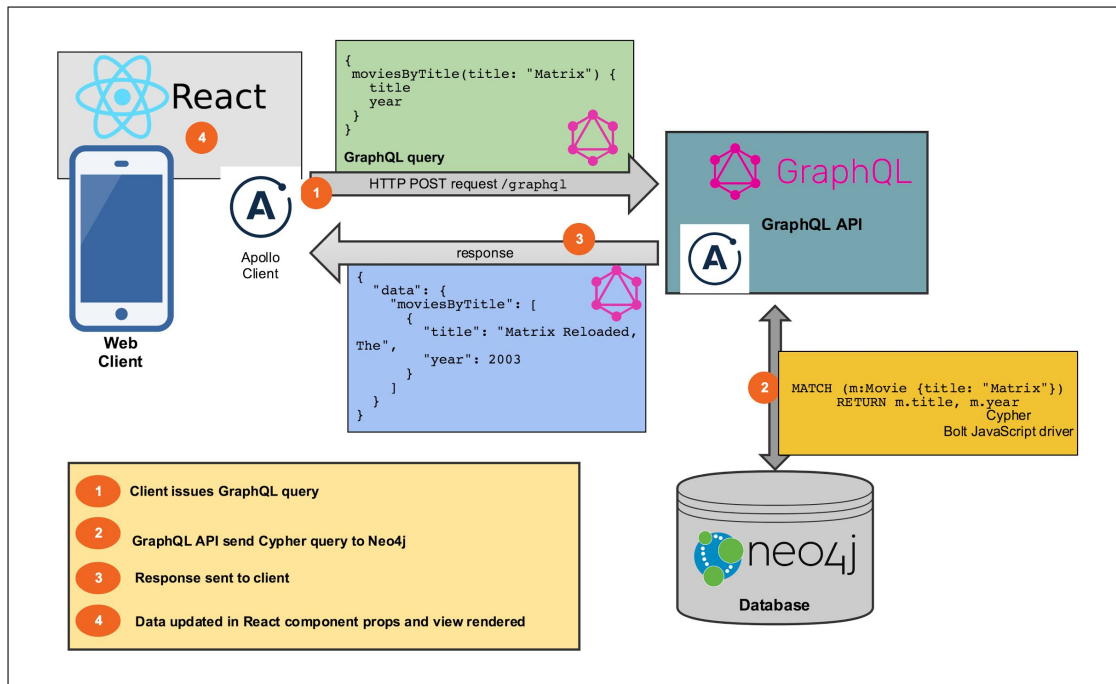
GraphQL



React



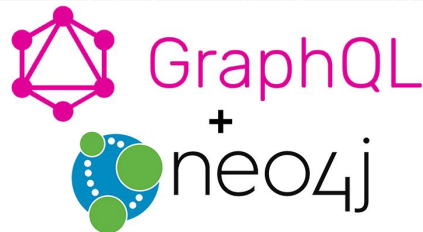
The GRAND stack



Looking Forward

- Active development
 - Feedback driven :-)
- Features
 - Subscriptions
 - Authentication / Authorization
 - ????

will@neo4j.com



neo4j.com/developer/graphql/

GRANDSTACK

grandstack.io/

(you)-[:HAVE]->(questions)<-[:ANSWERS]-(will)