

Methods of Importing into Neo4j

Sven Janko

sven.janko@neo4j.com

<https://neo4j.com/>



Agenda

- Recommended precognition
- Motivation Import
- Methods of importing
 - `$./bin/neo4j-admin import`
 - Cypher via LOAD CSV & APOC
 - Procedures
 - BatchInserter
 - Driver via BOLT

Recommended precognition

- Neo4j basics
 - Native, ACID compliant graph database
 - Property-graph model
 - Indexes / constraints
 - Installation, starting & stopping Neo4j
- Cypher and APOC basics
- Simple command line executions

<https://neo4j.com/>

Motivation

- Play with Neo4j
- Evaluation of a project idea
- PoC (Proof of Concept)
- Initializing the Neo4j graph at project kick-off
- Regular updates to the graph
- Eureka! That's how my data is connected



```
$ ./bin/neo4j-admin import
```



\$./bin/neo4j-admin import

- Fastest method (*w.r.t. writes/second*)
- Initial import; a new database is being created
- Database is offline during import
- No need to create indexes *in advance*
- The cluster needs to be synchronized after the import

Two Modes

--mode=CSV

```
personId:ID,name,:LABEL
```

```
keanu,"Keanu Reeves",Actor
```

```
:START_ID,role,:END_ID,:TYPE
```

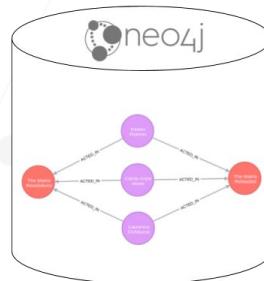
```
keanu,"Neo",tt0133093,ACTED_IN
```

```
keanu,"Neo",tt0234215,ACTED_IN
```

```
[...]
```

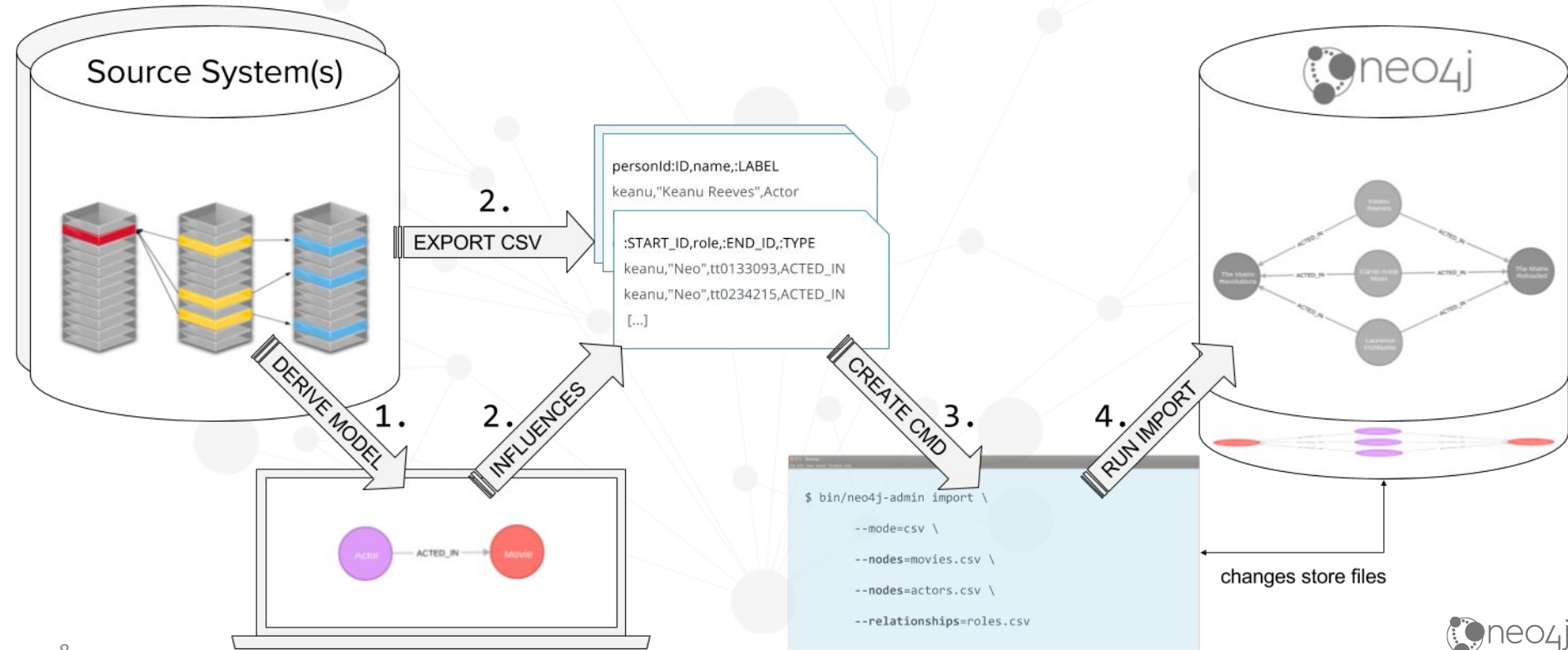
Import from CSV files

--mode=DATABASE



Import from a Neo4j graph

Import Approach



Our simple example schema



movies.csv (nodes)

movieId:**ID**,title,year:**int**,:**LABEL**

tt0133093,"The Matrix",1999,Movie

tt0234215,"The Matrix Reloaded",2003,**Movie;Sequel**

tt0242653,"The Matrix Revolutions",2003,**Movie;Sequel**

actors.csv (nodes)

personId:ID,name,:LABEL

keanu,"Keanu Reeves",Actor

laurence,"Laurence Fishburne",Actor

carrieanne,"Carrie-Anne Moss",Actor

roles.csv (relationships)

:START_ID,role,:END_ID,:TYPE

keanu,"Neo",tt0133093,ACTED_IN

keanu,"Neo",tt0234215,ACTED_IN

keanu,"Neo",tt0242653,ACTED_IN

laurence,"Morpheus",tt0133093,ACTED_IN

laurence,"Morpheus",tt0234215,ACTED_IN

[...]

Links between the CSV Files

personId:ID,name,:LABEL

- keanu,"Keanu Reeves",Actor
- laurence,"Laurence Fishburne",Actor
- carrieanne,"Carrie-Anne Moss",Actor

movieId:ID,title,year:int,:LABEL

- tt0133093,"The Matrix",1999,Movie
- tt0234215,"The Matrix Reloaded",2003,**Movie;Sequel**
- tt0242653,"The Matrix Revolutions",2003,**Movie;Sequel**

:START_ID,role,:END_ID,:TYPE

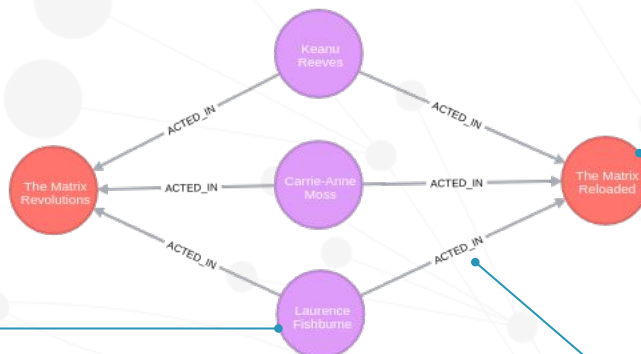
- keanu,"Neo",tt0133093,ACTED_IN
- keanu,"Neo",tt0234215,ACTED_IN
- keanu,"Neo",tt0242653,ACTED_IN
- laurence,"Morpheus",tt0133093,ACTED_IN
- laurence,"Morpheus",tt0234215,ACTED_IN
- [...]

The Import Command

```
Terminal
File Edit View Search Terminal Help

$ bin/neo4j-admin import \
  --mode=csv \
  --nodes=movies.csv \
  --nodes=actors.csv \
  --relationships=roles.csv
```

The Resulting Graph



personId:ID,name,:LABEL

keanu,"Keanu Reeves",Actor

laurence,"Laurence Fishburne",Actor

carrieanne,"Carrie-Anne Moss",Actor

movieId:ID,title,year,:LABEL

tt0133093,"The Matrix",1999,Movie

tt0234215,"The Matrix Reloaded",2003,Movie;Sequel

tt0242653,"The Matrix Revolutions",2003,Movie;Sequel

:START_ID,role,:END_ID,:TYPE

keanu,"Neo",tt0133093,ACTED_IN

keanu,"Neo",tt0234215,ACTED_IN

keanu,"Neo",tt0242653,ACTED_IN

laurence,"Morpheus",tt0133093,ACTED_IN

laurence,"Morpheus",tt0234215,ACTED_IN

[...]

Further Notes

- **Check the quality of your CSV files!**
- Separate header and data lines
- Use further options like...
 - additional-config
 - ignore-extra-columns
 - ignore-duplicate-nodes
 - ignore-missing-nodes
 - max-memory

Remember: Two Modes

--mode=CSV

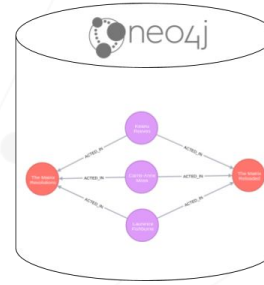


personId:ID,name,:LABEL
keanu,"Keanu Reeves",Actor

:START_ID,role,:END_ID,:TYPE
keanu,"Neo",tt0133093,ACTED_IN
keanu,"Neo",tt0234215,ACTED_IN
[...]

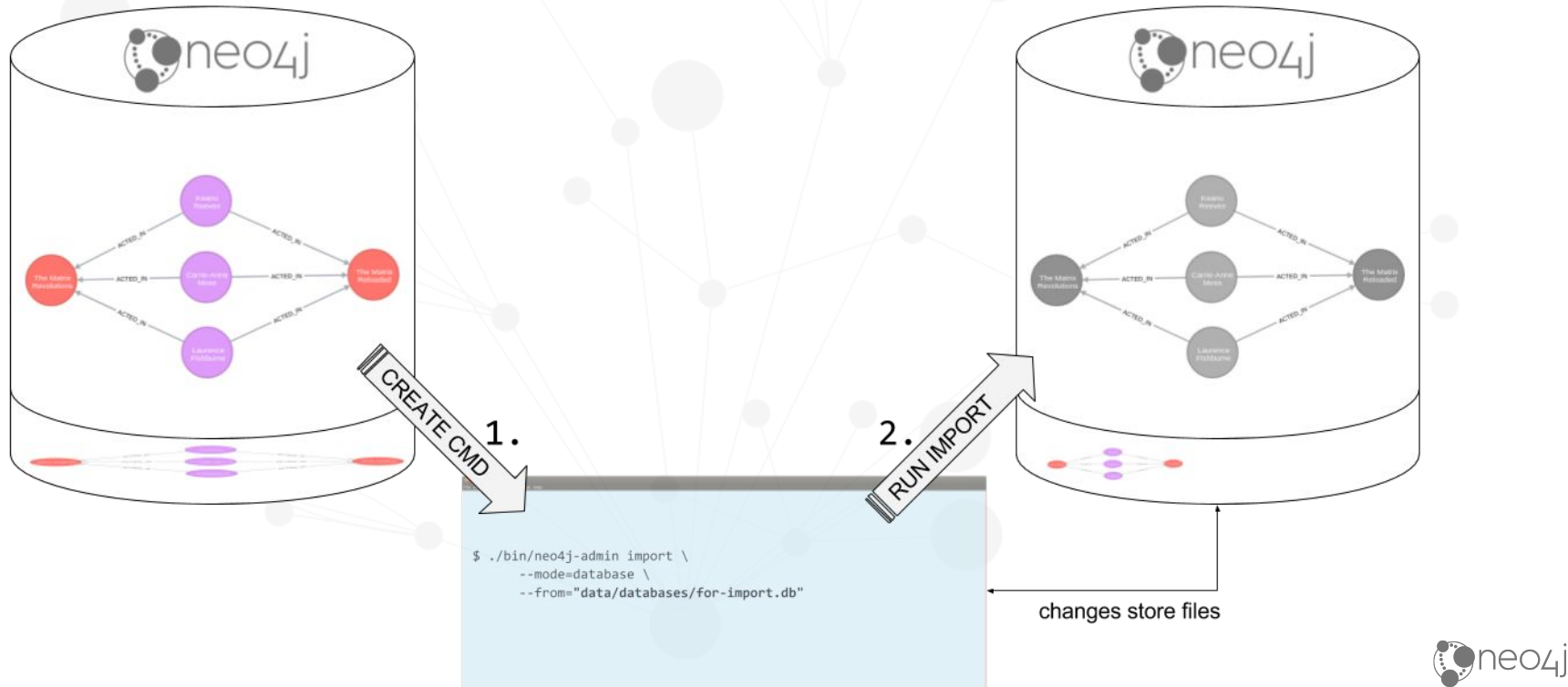
Import from CSV files

--mode=DATABASE



Import from a Neo4j graph

\$./bin/neo4j-admin import database



Mini Example

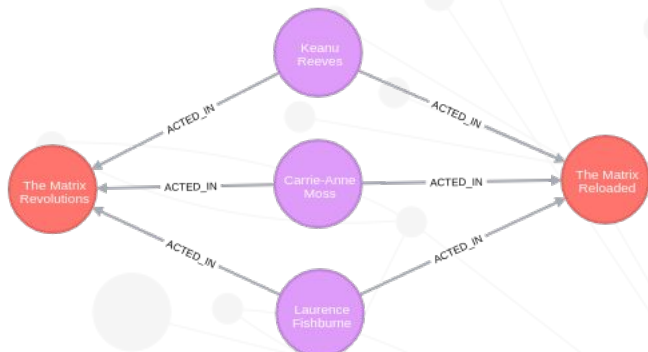
Terminal
File Edit View Search Terminal Help

```
$ ./bin/neo4j-admin import \  
  --mode=database \  
  --from="data/databases/for-import.db"
```

Path to Neo4j graph folder for
importing.

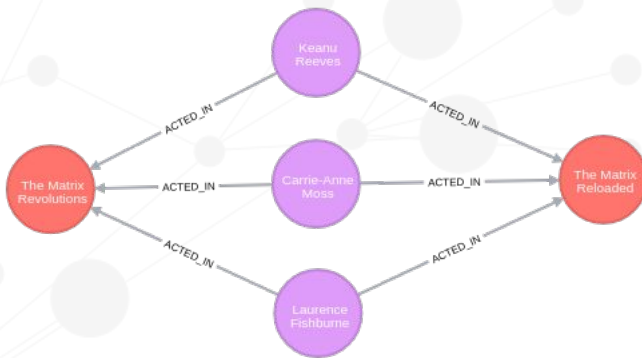
The Resulting Graph

Existing Neo4j graph



=

Newly imported Neo4j graph



`import --mode=database`

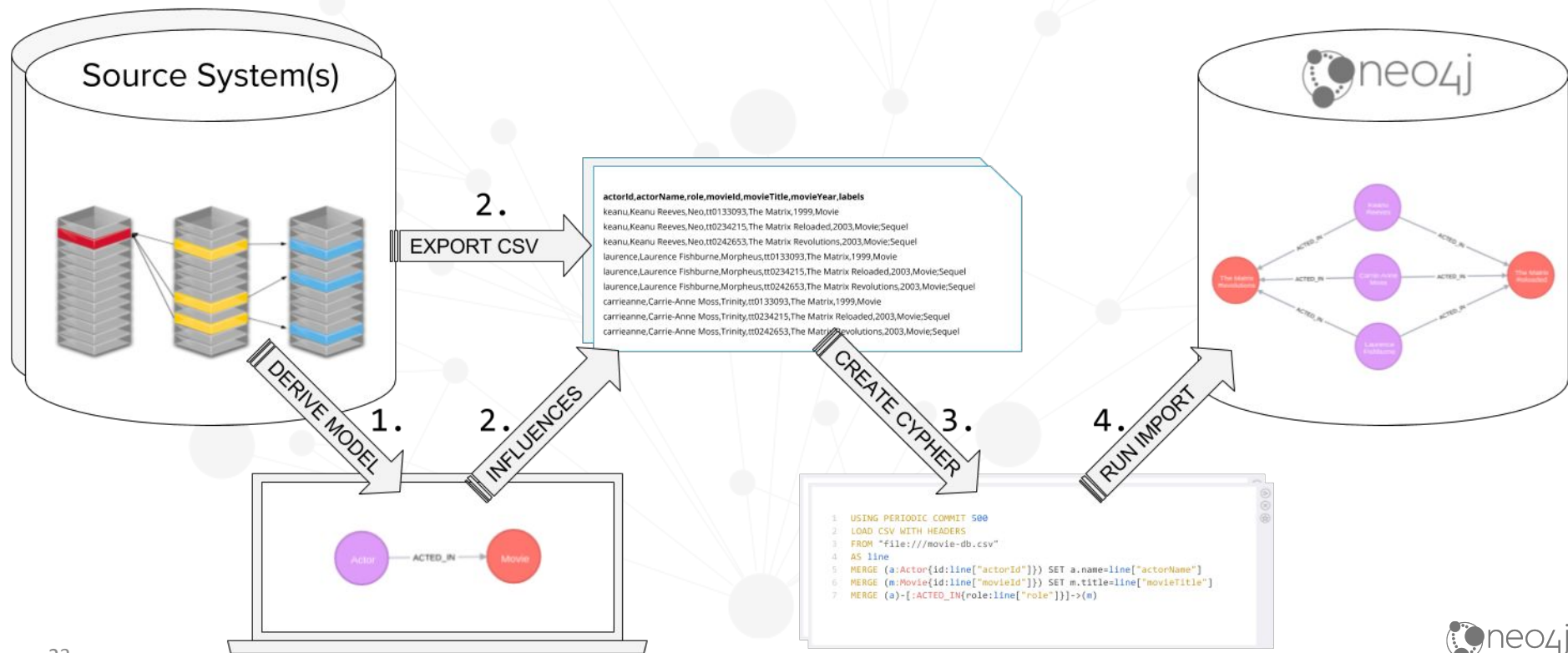
Cypher & LOAD CSV



Cypher & LOAD CSV

- May be the simplest method
- Initial import or update
- Database is online during import, transactional!
- Create indexes upfront
- The cluster is being synchronized automatically

Cypher (LOAD CSV)



movie-db.csv (nodes & relationships)

actorId,actorName,role,movieId,movieTitle,movieYear,labels

keanu,Keanu Reeves,Neo,tt0133093,The Matrix,1999,Movie

keanu,Keanu Reeves,Neo,tt0234215,The Matrix Reloaded,2003,Movie;Sequel

keanu,Keanu Reeves,Neo,tt0242653,The Matrix Revolutions,2003,Movie;Sequel

laurence,Laurence Fishburne,Morpheus,tt0133093,The Matrix,1999,Movie

laurence,Laurence Fishburne,Morpheus,tt0234215,The Matrix Reloaded,2003,Movie;Sequel

laurence,Laurence Fishburne,Morpheus,tt0242653,The Matrix Revolutions,2003,Movie;Sequel

carrieanne,Carrie-Anne Moss,Trinity,tt0133093,The Matrix,1999,Movie

carrieanne,Carrie-Anne Moss,Trinity,tt0234215,The Matrix Reloaded,2003,Movie;Sequel

carrieanne,Carrie-Anne Moss,Trinity,tt0242653,The Matrix Revolutions,2003,Movie;Sequel

LOAD CSV - Initial Test

```
1 LOAD CSV WITH HEADERS
2 FROM "file:///movie-db.csv"
3 AS line
4 RETURN line
5 LIMIT 1
```

Relative to the *import* folder
of the Neo4j installation

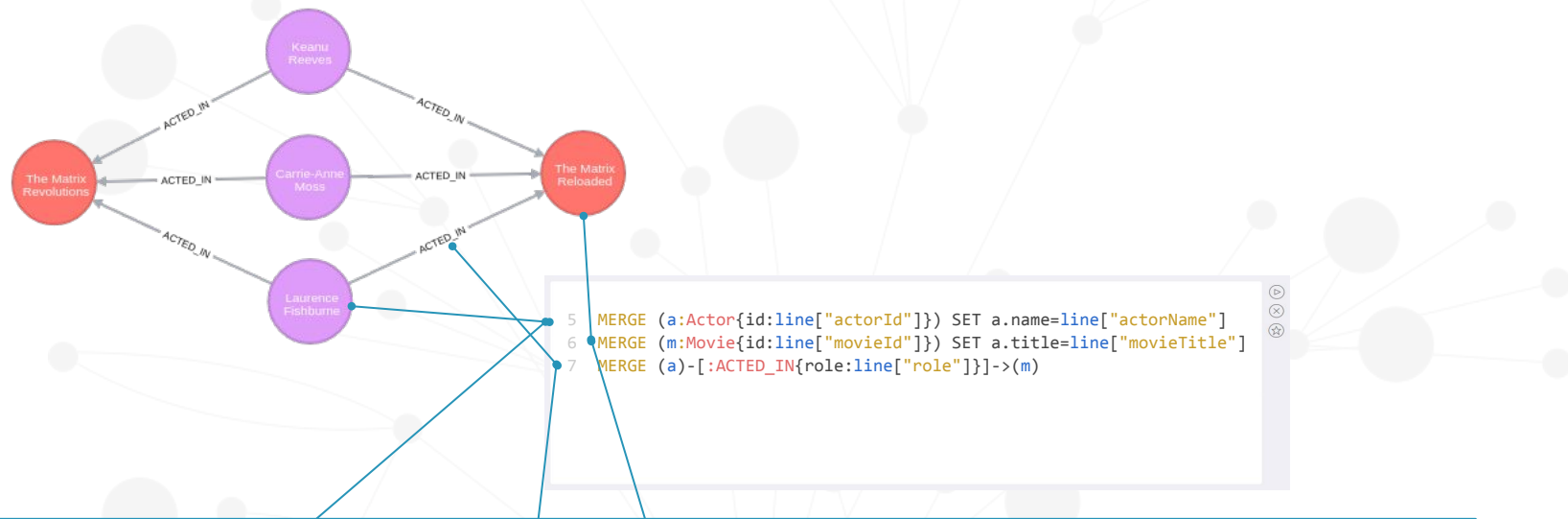
(if not changed in neo4j.conf)

```
{
  "role": "Neo",
  "actorName": "Keanu Reeves",
  "actorId": "keanu",
  "labels": "Movie",
  "movieId": "tt0133093",
  "movieYear": "1999",
  "movieTitle": "The Matrix"
}
```

LOAD CSV - Mini Example

```
1  USING PERIODIC COMMIT 500
2  LOAD CSV WITH HEADERS
3  FROM "file:///movie-db.csv"
4  AS line
5  MERGE (a:Actor{id:line["actorId"]}) SET a.name=line["actorName"]
6  MERGE (m:Movie{id:line["movieId"]}) SET m.title=line["movieTitle"]
7  MERGE (a)-[:ACTED_IN{role:line["role"]}]->(m)
```

The Resulting Graph



actorId,actorName,role,movieId,movieTitle,movieYear,labels

keanu,Keanu Reeves,Neo,tt0133093,The Matrix,1999,Movie

[..]

laurence,Laurence Fishburne,Morpheus,tt0234215,The Matrix Reloaded,2003,Movie;Sequel

Further Notes

- Check the quality of your CSV files!
- Create indexes
- Avoid the *Eager* operator, e.g.
 - With **PROFILE LOAD CSV** ... or
 - Check for warnings in the Neo4j Browser
- Choose size of your transactions, e.g. by
 - **USING PERIODIC COMMIT** or
 - **CALL apoc.periodic.iterate**

Cypher & APOC



Cypher & APOC

- Iterate / batching
- Plenty of procedures and functions
- GraphML
- JDBC
- .. and others (e.g. XML, JSON, ...)

apoc.periodic.iterate

```
1
2  LOAD CSV WITH HEADERS
3  FROM "file:///movie-db.csv"
4  AS line
5  MERGE (a:Actor{id:line["actorId"]}) SET a.name=line["actorName"]
6  MERGE (m:Movie{id:line["movieId"]}) SET m.title=line["movieTitle"]
7  MERGE (a)-[:ACTED_IN{role:line["role"]}]->(m)
8
9
```

apoc.periodic.iterate

```
1 CALL apoc.periodic.iterate(  
2   'LOAD CSV WITH HEADERS  
3   FROM "file:///movie-db.csv"  
4   AS line RETURN line',  
5   'MERGE (a:Actor{id:line["actorId"]}) SET a.name=line["actorName"]  
6   MERGE (m:Movie{id:line["movieId"]}) SET m.title=line["movieTitle"]  
7   MERGE (a)-[:ACTED_IN{role:line["role"]}]->(m)',  
8   {batchSize:500}  
9 )
```



movie-db.csv

actorId,actorName,role,movieId,movieTitle,movieYear,**labels**

keanu,Keanu Reeves,Neo,tt0133093,The Matrix,1999,**Movie**

keanu,Keanu Reeves,Neo,tt0234215,The Matrix Reloaded,2003,**Movie;Sequel**

keanu,Keanu Reeves,Neo,tt0242653,The Matrix Revolutions,2003,**Movie;Sequel**

laurence,Laurence Fishburne,Morpheus,tt0133093,The Matrix,1999,**Movie**

laurence,Laurence Fishburne,Morpheus,tt0234215,The Matrix Reloaded,2003,**Movie;Sequel**

laurence,Laurence Fishburne,Morpheus,tt0242653,The Matrix Revolutions,2003,**Movie;Sequel**

carrieanne,Carrie-Anne Moss,Trinity,tt0133093,The Matrix,1999,**Movie**

carrieanne,Carrie-Anne Moss,Trinity,tt0234215,The Matrix Reloaded,2003,**Movie;Sequel**

carrieanne,Carrie-Anne Moss,Trinity,tt0242653,The Matrix Revolutions,2003,**Movie;Sequel**

apoc.create.addLabels

```
1  LOAD CSV WITH HEADERS
2  FROM "file:///movie-db.csv"
3  AS line
4  MATCH (m:Movie{id:line["movieId"]})
5  CALL apoc.create.addLabels(m, split(line["labels"], ";"))
6  YIELD node RETURN node
```

Import GraphML

in neo4j.conf:

```
apoc.import.file.enabled=true
```

```
1 CALL apoc.import.graphml(  
2   "graphml.xml",  
3   { batchSize: 5000, readLabels: true }  
4 )
```

```
1 <?xml version="1.0" encoding="UTF-8"?>  
2 <graphml xmlns="http://graphml.graphdrawing.org/xmlns" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns  
3 http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">  
4 <graph id="G" edgedefault="directed">  
5   <node id="n6" labels=":Actor">  
6     <data key="id">keanu</data><data key="name">Keanu Reeves</data>  
7   </node>  
8   <node id="n7" labels=":Movie">  
9     <data key="id">tt0133093</data><data key="title">The Matrix</data>  
10  </node>  
11  <edge id="e3" source="n6" target="n7" label="ACTED_IN"><data key="role">Neo</data></edge>  
12 </graph>  
13 </graphml>
```

Import via JDBC

Table or SQL statement

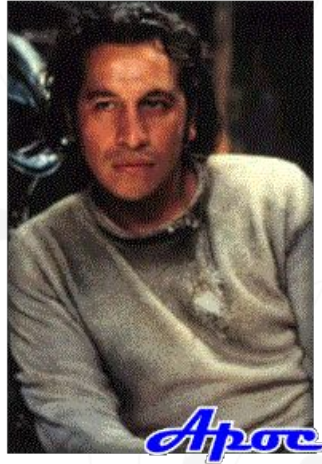
```
1 CALL apoc.load.jdbc('jdbc:mysql://host:3306/mdb?user=test','movies')
2 YIELD row
3 CREATE (:Movie {title:row.title, year:row.year})
```

title	year
The Matrix	1999
The Matrix Reloaded	2003
The Matrix Revolutions	2003

URL can be defined in neo4j.conf

So that user/password is not part of the Cypher query

APOC User Guide



<https://neo4j-contrib.github.io/neo4j-apoc-procedures/>

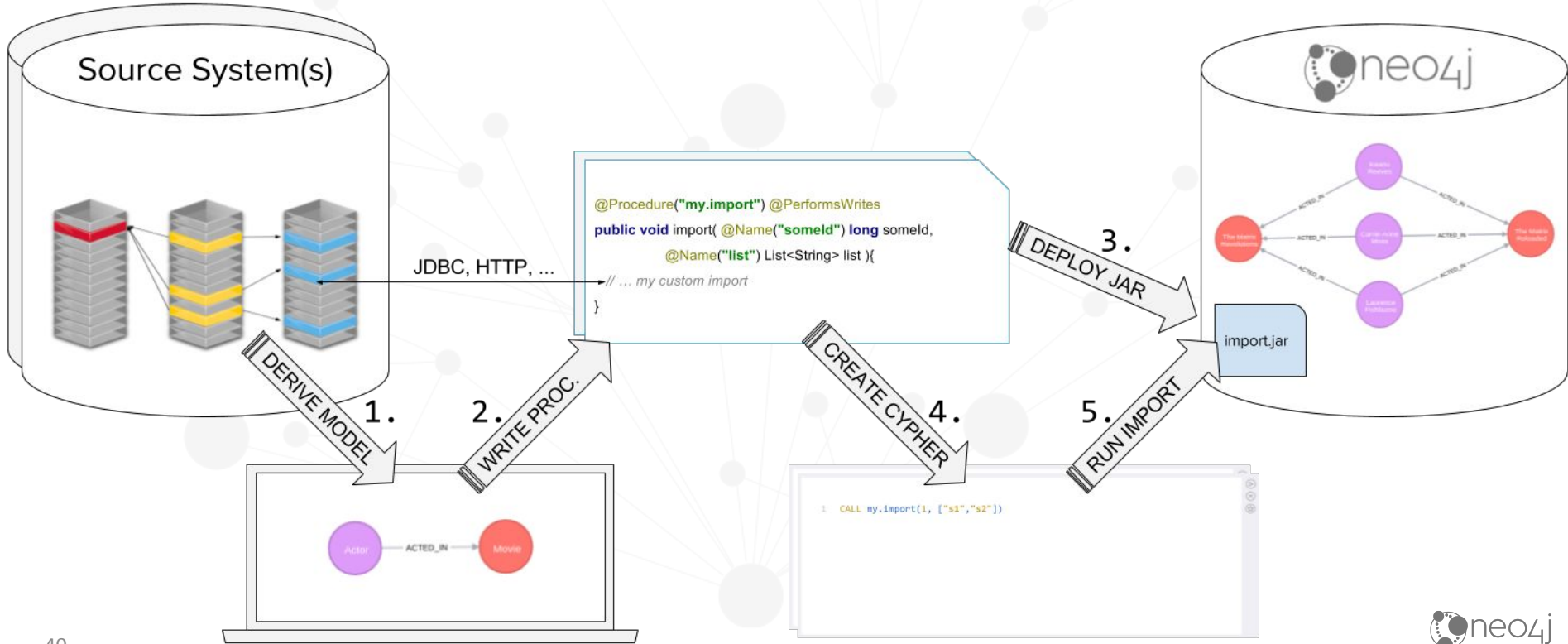
Procedures



Procedures

- Extension of the Neo4j server
 - Will be deployed as .jar file to the plugins folder
- Database is online during import, transactional!
- Make use of one of our APIs for graph processing
 - => Performance
- Fine grained user/role concept
- The cluster is being synchronized automatically

Procedures



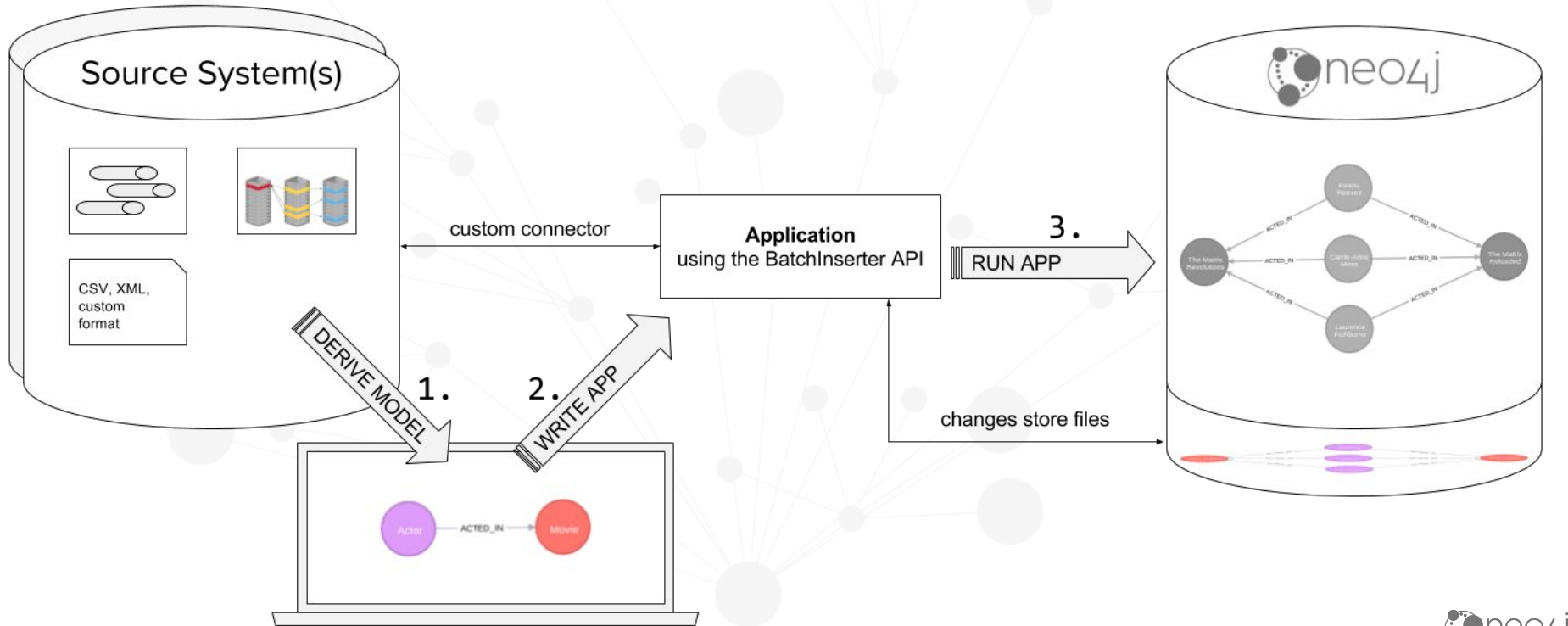
BatchInserter



BatchInserter

- Initial import or update
- Not transactional! Not thread-safe! Private API!
- But extremely fast
- Database is offline during import
- Can handle complex data transformations
- The cluster needs to be synchronized after the import

BatchInserter



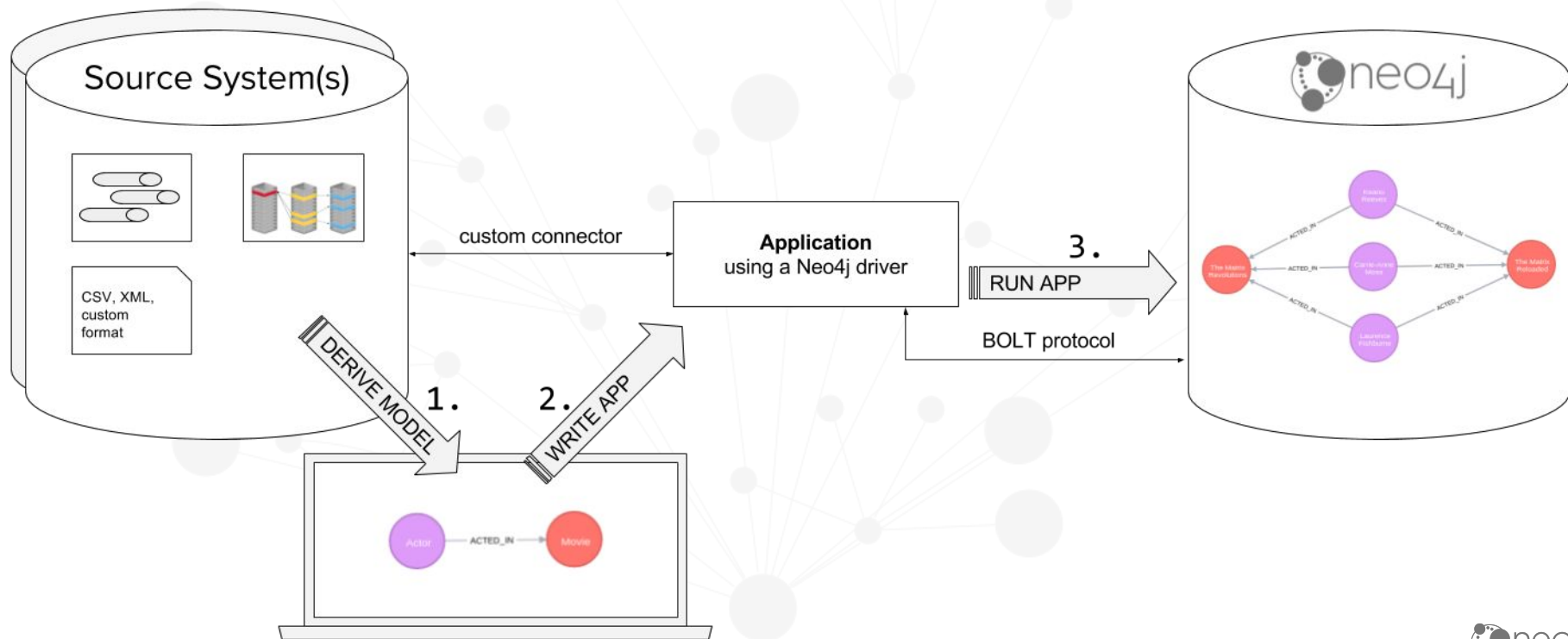
Driver via BOLT




Driver via BOLT

- Drivers for many languages available
- Transactional processing
- Batching
- Parallelization possible

Driver via BOLT

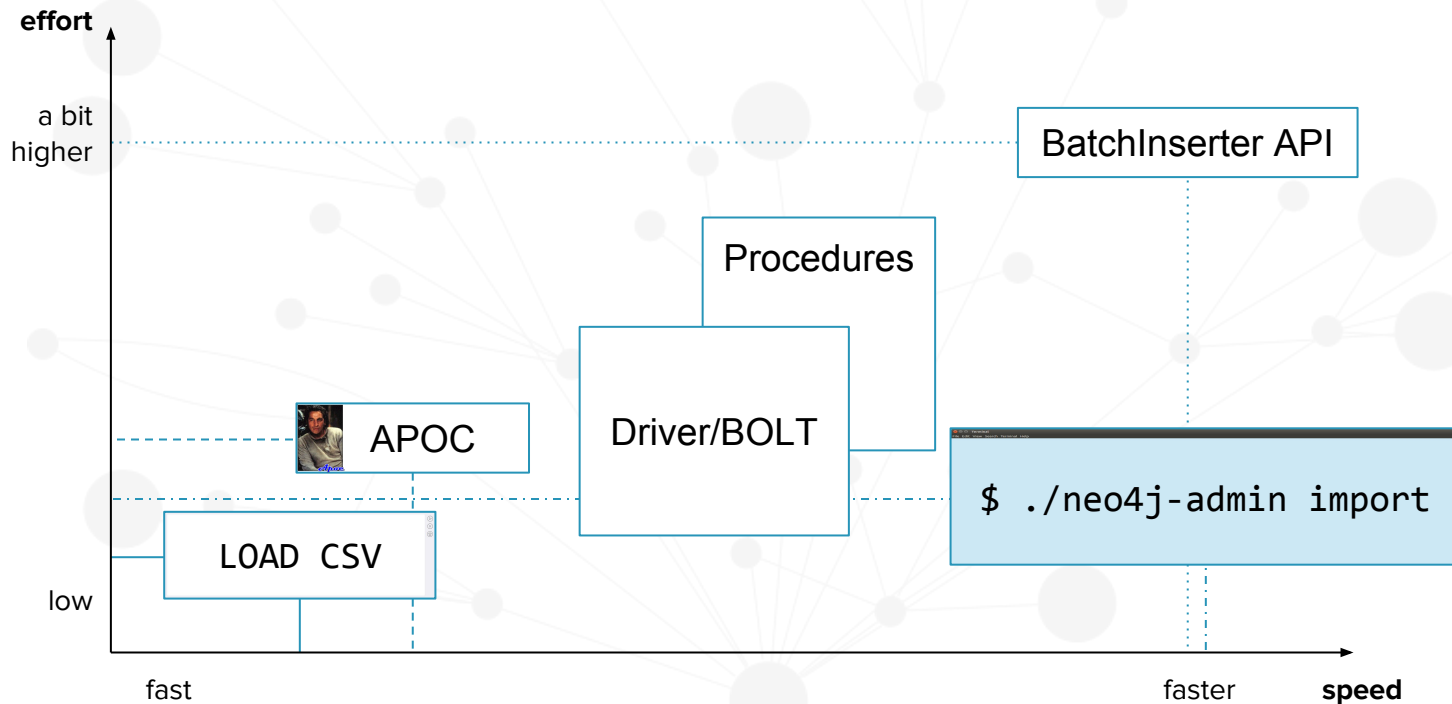




So, which method should I
use?



It depends...





Documentation

\$./neo4j-admin import - <https://neo4j.com/docs/operations-manual/current/tools/import/>

LOAD CSV

- <http://neo4j.com/docs/developer-manual/current/cypher/clauses/load-csv/>
- <https://neo4j.com/developer/guide-import-csv/>

Eager operator

- <https://neo4j.com/docs/developer-manual/current/cypher/execution-plans/row-operators/#query-plan-eager>
- <http://www.markneedham.com/blog/2014/10/23/neo4j-cypher-avoiding-the-eager/>

APOC - <https://neo4j-contrib.github.io/neo4j-apoc-procedures/>

Procedures - <http://neo4j.com/docs/developer-manual/current/extending-neo4j/procedures/>

BatchInserter - <http://neo4j.com/docs/java-reference/current/javadocs/org/neo4j/unsafe/batchinsert/BatchInserter.html>

Drivers

- <https://neo4j.com/docs/developer-manual/current/drivers/>
- <https://neo4j.com/developer/language-guides/>