

# Neo4j 3.0

Benoit Simard (@logisima)



# introduction

# Benoit Simard, consultant



- **[benoit.simard@neotechnology.com](mailto:benoit.simard@neotechnology.com)**
- @logisima
- Addicte aux graphes
- Formateur

# Les prochains évènements



- **1er juin** : Les graphes dans les projets industriels orientés data, par Kernix
- **15 juin** : Venue sur Paris de Andres Taylor (le créateur de Cypher)
- **28 juin**: Graphes & Finance - Plongée dans les Panama Papers
- **21 septembre** : Graph day Paris

# L'histoire de Neo4j

# Neo4j 0.X - 2000 à 2010



- API Java
- JAR de petite taille
- Indexation basique

# Neo4j 1.X - 2010 à 2014



- API REST
- Serveur Neo4j
- Mode cluster
- Cypher dans la 1.4

# Neo4j 2.X - 2014 à 2016



- Cypher par défaut (ajout de l'écriture)
- Ajout des Labels
- Schema
- Neo4j Browser
- Page Cache

# Neo4j 3.0

# Les fondements

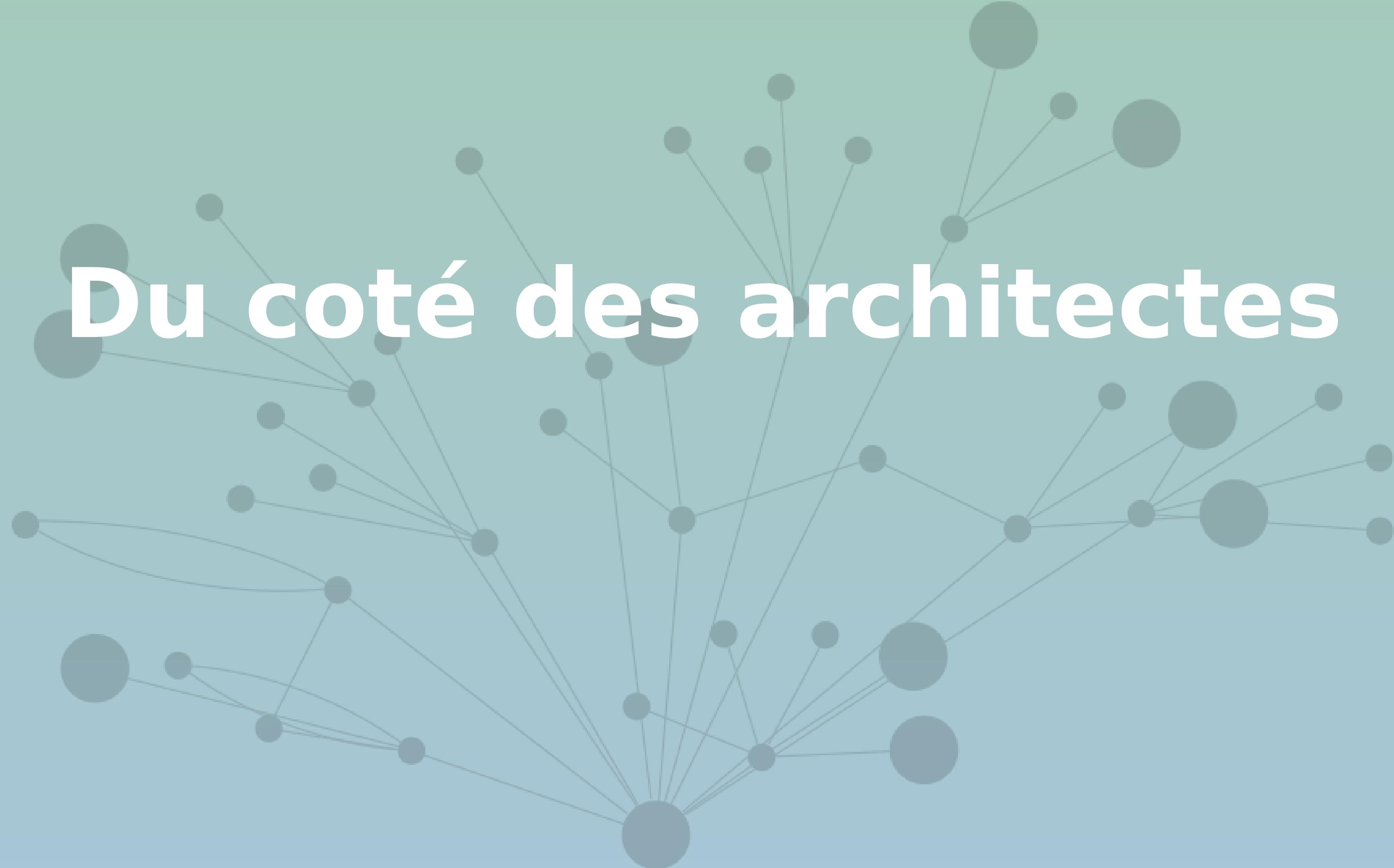


Permettre à tout le monde de développer des applications graphes

- Simplement
- Rapidement
- Sans contrainte de volumétrie

# Les fondements





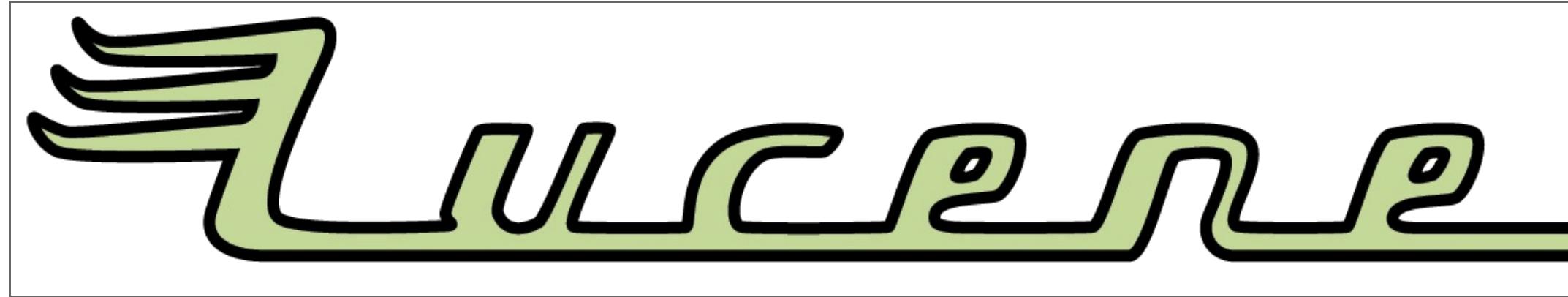
# Du côté des architectes

# Jusqu'au bout de l'extrême limite !

**EE** : Un nouveau moteur de stockage qui abolit les limites de Neo4j (> 1 quadrillion) !



# Les indexes, c'est majeur



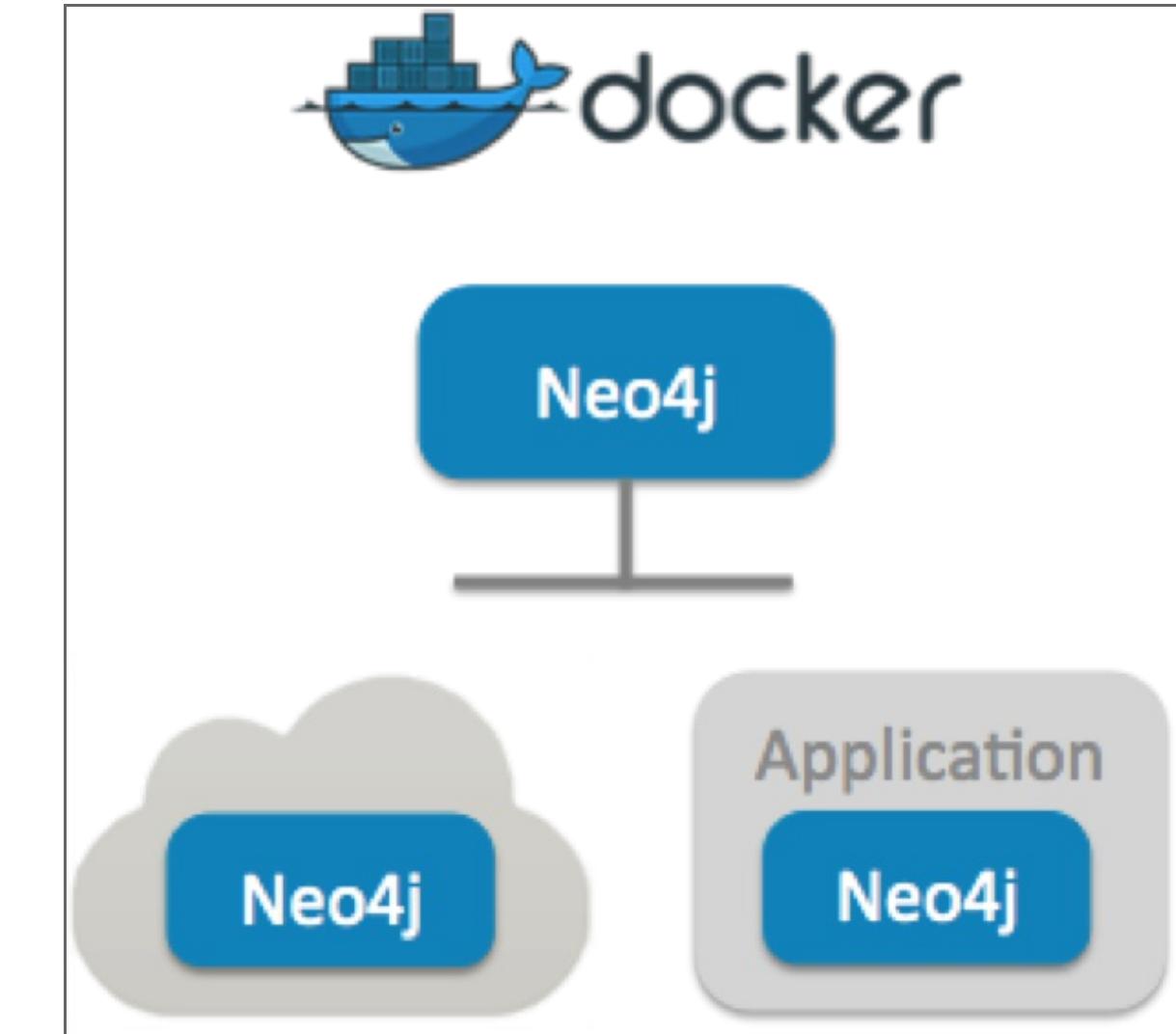
- Mise à jour de lucene en version 5
- Indexation en parallèle (meilleure performance en écriture)
- Partitonnement des indexes lucene (suppression de la limitation de 2 milliards)



**Pour les ops !**

# Cloud , virtualisation, containers

- Amélioration des performances sur les environnements virtualisés (Optimisation du page-cache)
- Image docker officielle



# Refactoring : la structure

Modification de la structure

# Refactoring : la configuration

Un seul fichier de conf conf/neo4j.conf, avec les propriétés namespacés

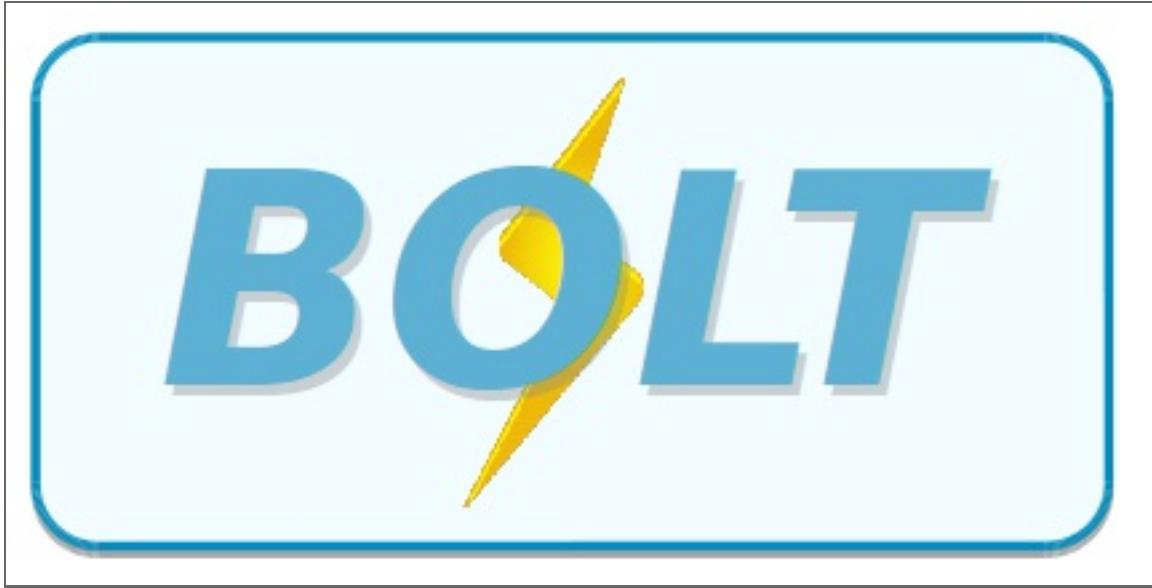
**/!\ Les noms des propriétés ont donc changé !**

# Refactoring : les logs

Tous les fichiers de logs sont à présent dans le répertoire logs

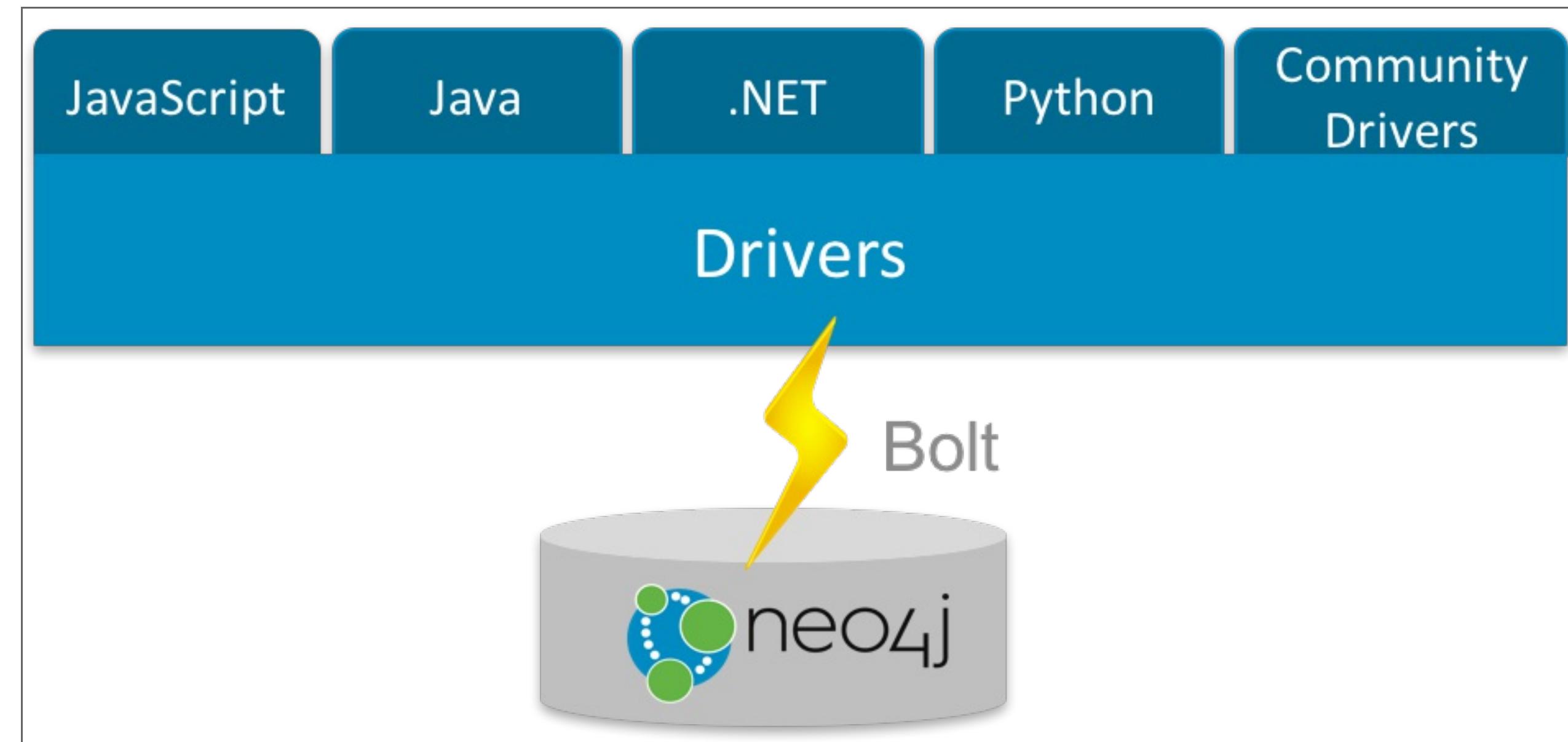
# Le coin des développeurs

# Bolt



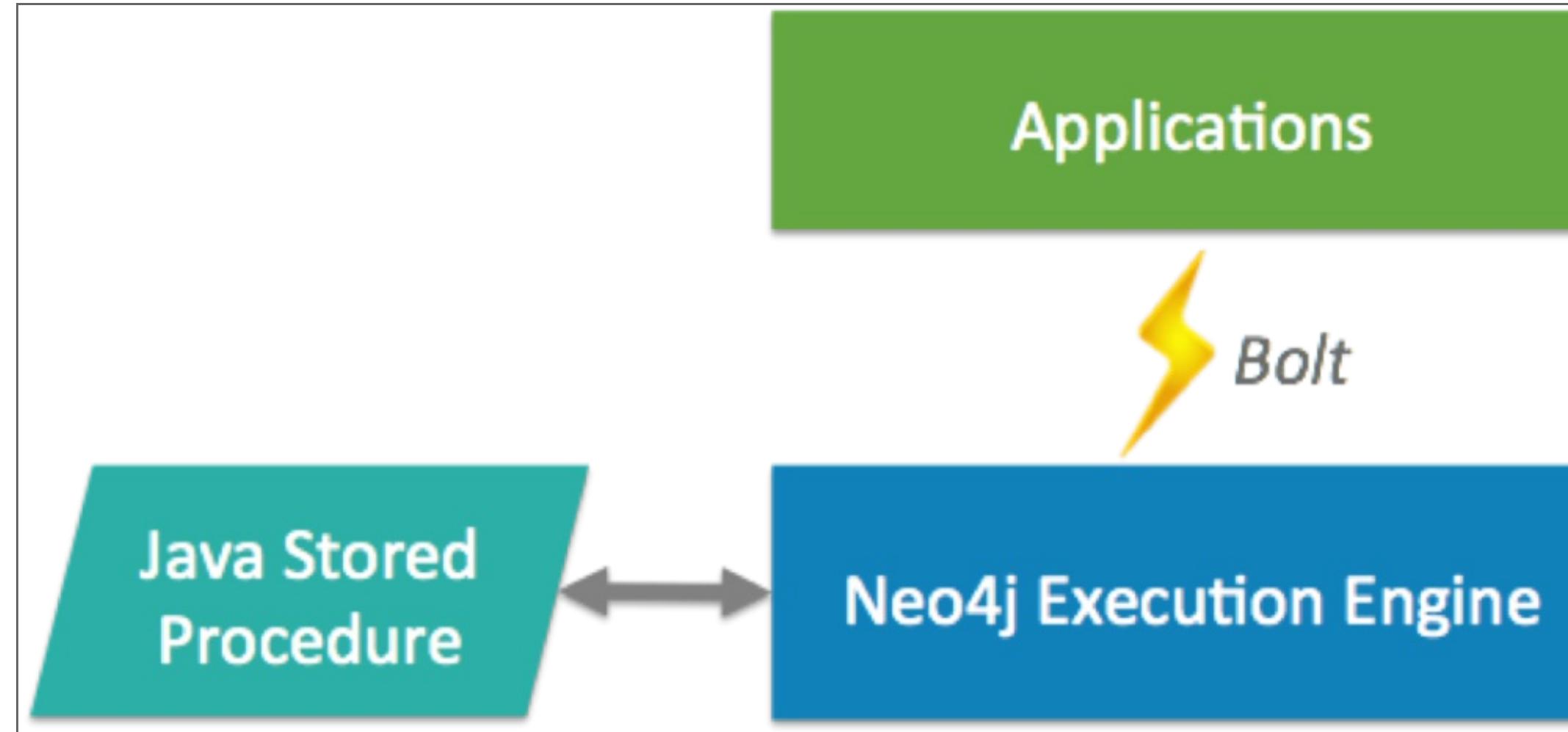
- **Protocole binaire**
- **Transport** : TCP & Websocket
- **Sécurisé** : TLS & authentification par défaut
- Versionné
- bolt://localhost:7687

# Bolt: drivers officiels



# Bolt

# Les procédures stockées



Ecrivez du code personnalisé :

- En Java (compatible JVM)
- Stocké sur le serveur (un jar à déployer)
- Accessible en Cypher

# Les procédures stockées : built-in

```
$ CALL dbms.procedures() YIELD name with split(name,".") AS parts WHERE not parts[0] = 'apoc' RETURN parts[0..-1] AS package, count(*),collect(parts[-1]) AS procs
```

Rows	package	count(*)	procs
[db]	5		[constraints, indexes, labels, propertyKeys, relationshipTypes]
[dbms]	4		[changePassword, components, procedures, queryJmx]

Returned 2 rows in 934 ms.

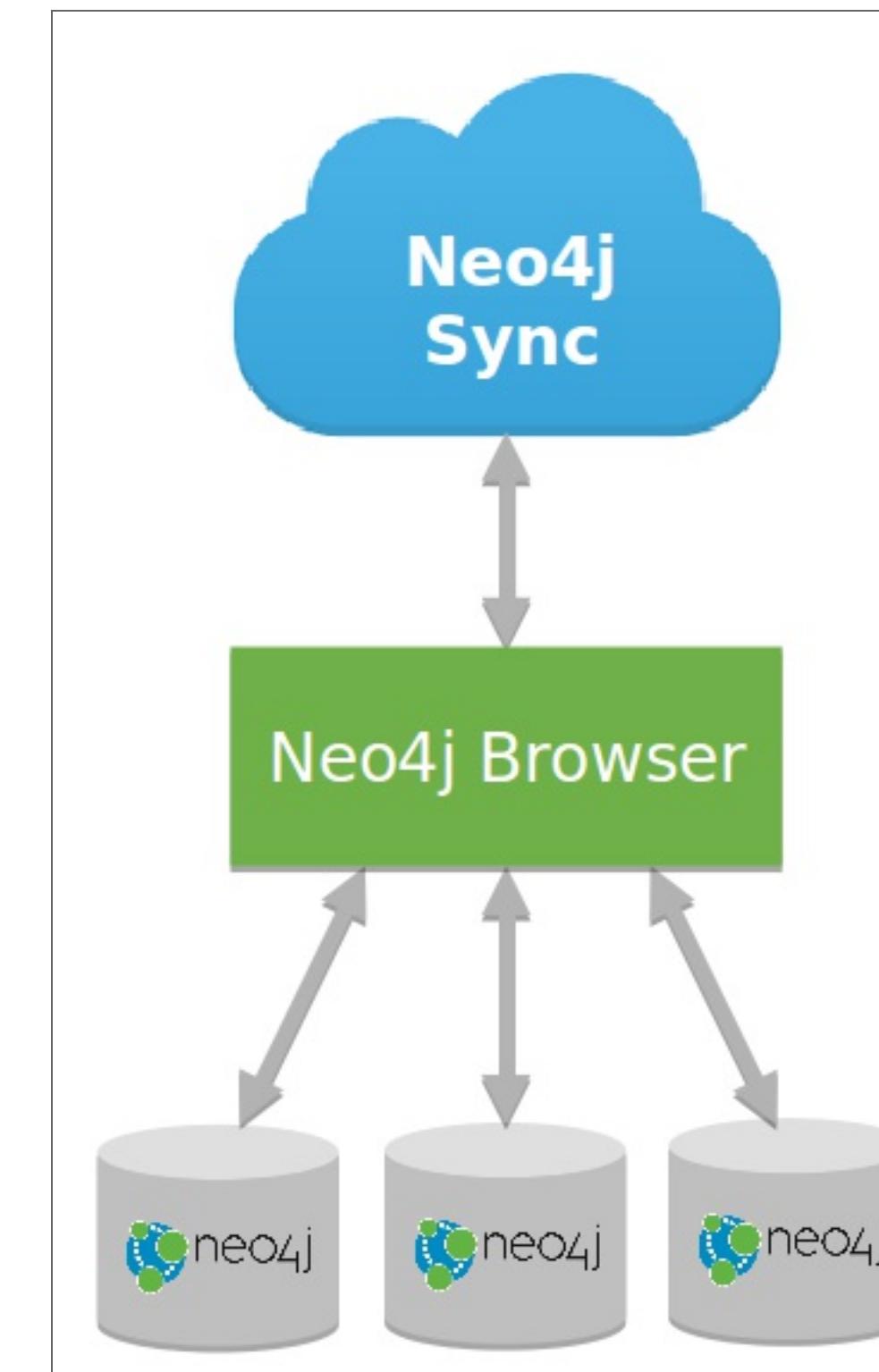
# APOC

[\*\*https://github.com/neo4j-contrib/neo4j-apoc-procedures\*\*](https://github.com/neo4j-contrib/neo4j-apoc-procedures)

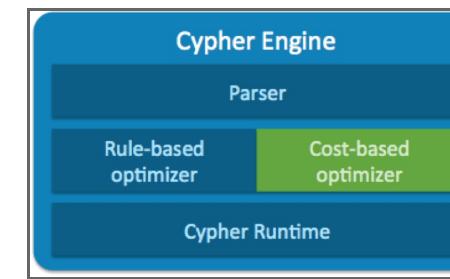
- Obtenir le meta-graph : CALL apoc.meta.graph()
- Charger des données en JDBC :
  - CALL apoc.load.driver('com.mysql.jdbc.Driver')
  - CALL apoc.load.jdbc('jdbc:mysql:localhost/mysql', 'SELECT \* FROM user') YIELD row CREATE (:User {name:row.User})
- Charger des données JSON
- Des fonctions spatiales
- ...

# Neo4j Browser Sync

Synchronisez vos scripts, configuration, style sur tous vos navigateurs



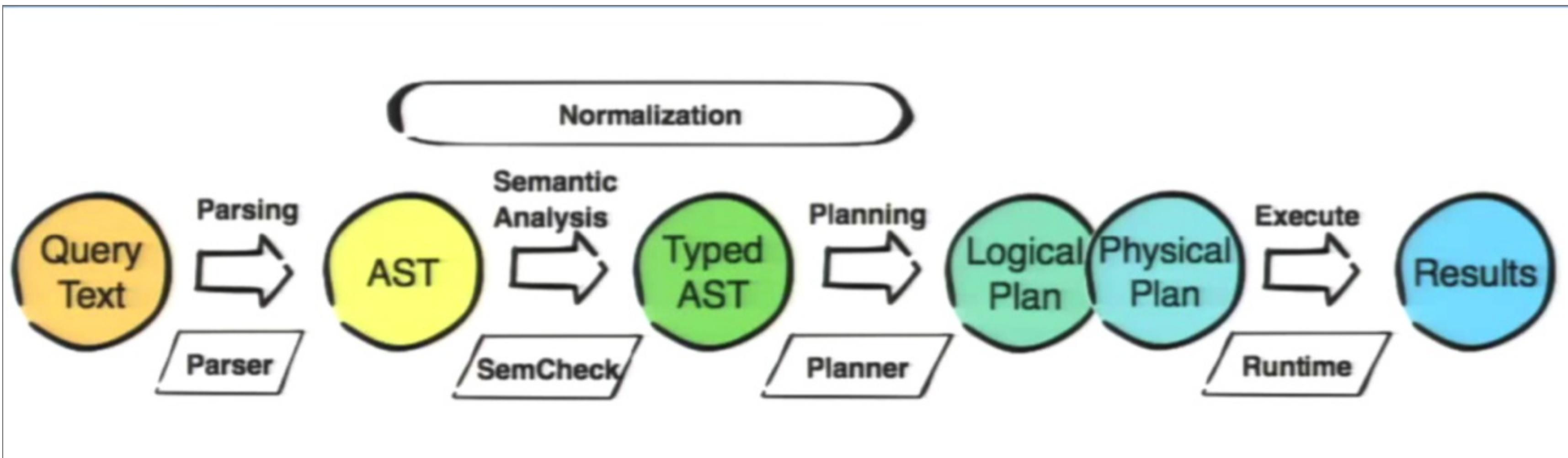
# Cypher



- ENDS WITH et CONTAINS se basent maintenant sur les indexes (tout comme STARTS WITH en 2.3)
- Amélioration globale des performances sur les aggregations (ex: count sur les labels)
- Amélioration des performances d'écritures / écritures
  - Upgrade de lucene
  - Planneur basé sur les coûts pour les écritures

# Cypher : planneur basé sur les coûts

- Basé sur la connaissance des données en base (ie. des statistiques)
- A été introduit en 2.2 pour les requêtes en lecture seule
- Est disponible à présent aussi pour les requêtes d'écritures
- Planneur par défaut



# Cypher : example

Cypher version: CYpher 3.0, planner: RULE. 3060001 total db hits in 7197 ms.

Cypher version: CYpher 3.0, planner: COST. 50003 total db hits in 3280 ms.

# Cypher : shortestpath

Les prédictats utilisés dans la clause WHERE sont maintenant évalués pendant l'algorithme !

# Comment upgrader ?

# Upgrader en 3.X

**<http://neo4j.com/guides/upgrade/#neo4j-3-0>**

- Java 8
- Mise à jour du store : dbms.allow\_format\_migration=true
- Mise à jour de lucene : reconstruction des indexes
- Mise à jour de la configuration : java -jar config-migrator.jar path/to/neo4j2.3 path/to/neo4j3.0

# Merci

# Des questions ?

- **Twitter:** Suivez les comptes @neojFr & @neo4j
- **Google group :** Avec les groupes Neo4jFr & Neo4j
- **Stackoverflow :** avec les tags neo4j & cypher
- **Slack :** <http://neo4j-users-slack-invite.herokuapp.com/>

