DS 4300

Neo4j

Mark Fontenot, PhD
Northeastern University

Neo4j

- A Graph Database System that supports both transactional and analytical processing of graph-based data
  - Relatively new class of no-sql DBs
- Considered schema optional (one can be imposed)
- Supports various types of indexing
- ACID compliant
- Supports distributed computing
- Similar: Microsoft CosmoDB, Amazon Neptune

Neo4j - Query Language and Plugins

- Cyper
  - Neo4j's graph query language created in 2011
  - Goal: SQL-equivalent language for graph databases
  - Provides a visual way of matching patterns and relationships `(nodes)-[:CONNECT_TO]->(otherNodes)`
- APOC Plugin
  - Awesome Procedures on Cypher
  - Add-on library that provides hundreds of procedures and functions
- Graph Data Science Plugin
  - provides efficient implementations of common graph algorithms (like the ones we talked about yesterday)

Neo4j in Docker Compose

Docker Compose

- Supports **multi-container management**.
- Set-up is declarative - using YAML docker-compose.yaml file
  - services
  - volumes
  - networks, etc.
- 1 command can be used to start, stop, or scale a number of services at one time.
- Provides a consistent method for producing an identical environment (no more “well… it works on my machine!)
- Interaction is mostly via command line

docker-compose.yaml

```
services:
  neo4j:
    container_name: neo4j
    image: neo4j:latest
    ports:
      - 7474:7474
      - 7687:7687
    environment:
      - NEO4J_AUTH=neo4j/${NEO4J_PASSWORD}
      - NEO4J_apoc_export_file_enabled=true
      - NEO4J_apoc_import_file_enabled=true
      - NEO4J_apoc_import_file_use__neo4j__config=true
      - NEO4J_PLUGINS=["apoc", "graph-data-science"]
    volumes:
      - ./neo4j_db/data:/data
      - ./neo4j_db/logs:/logs
      - ./neo4j_db/import:/var/lib/neo4j/import
      - ./neo4j_db/plugins:/plugins
```

Never put “secrets” in a docker compose file. Use .env files.

.env Files

- **.env** files - stores a collection of environment variables
- good way to keep environment variables for different platforms separate
  - .env.local
  - .env.dev
  - .env.prod

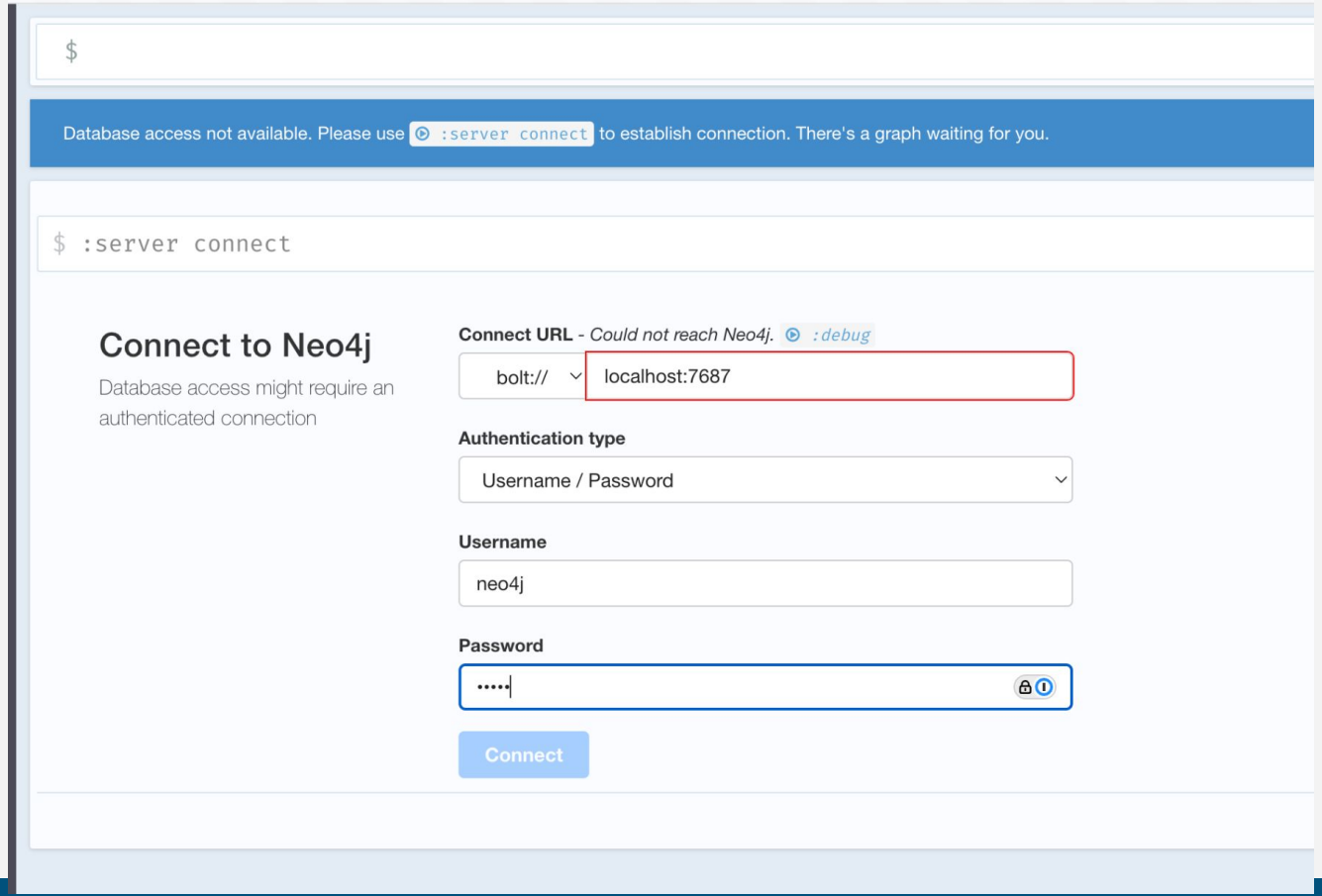
.env file

```
NEO4J_PASSWORD=abc123!!!
```

Docker Compose Commands

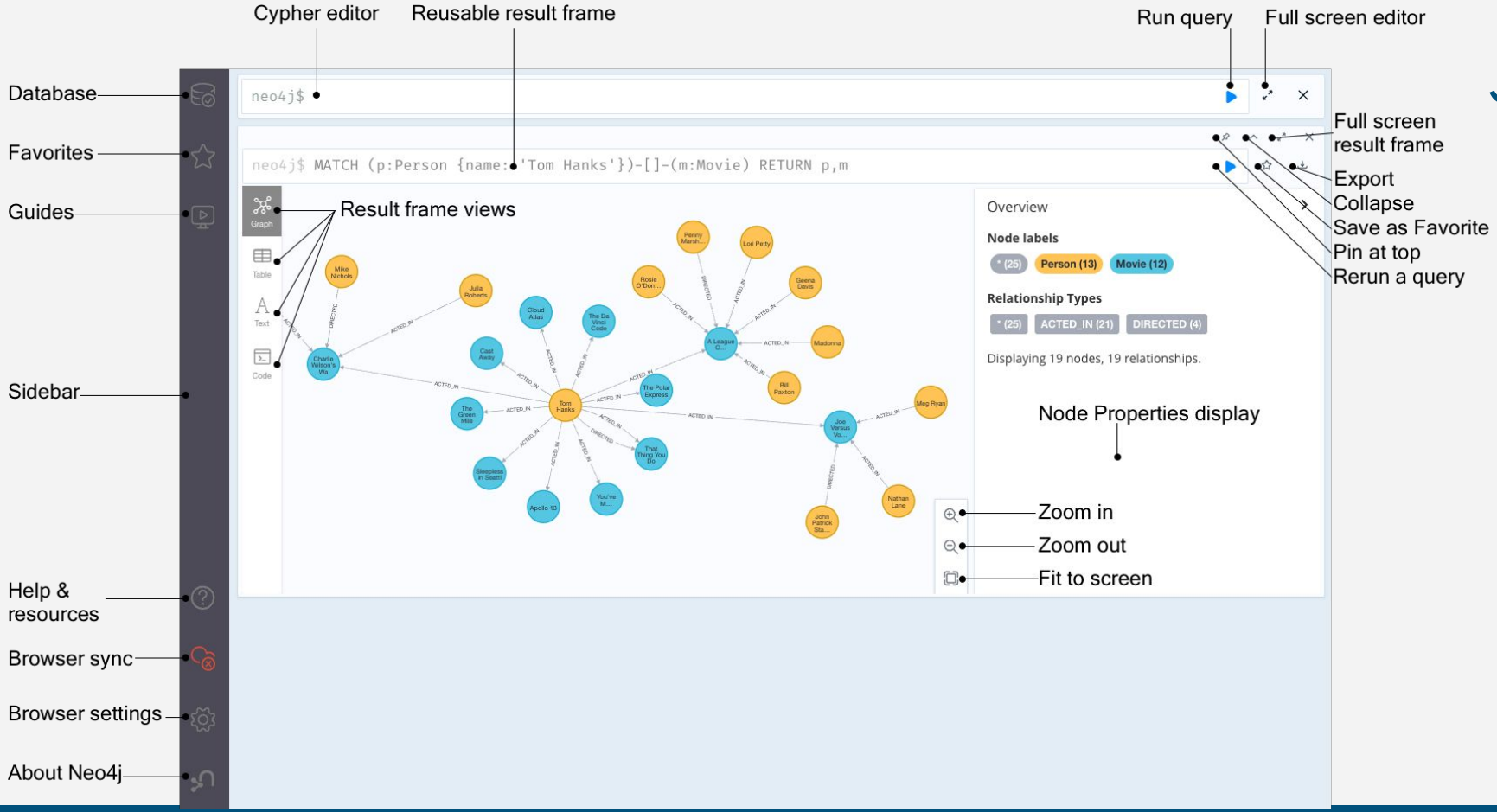
- To test if you have Docker CLI properly installed, run: docker --version
- Major Docker Commands
  - docker compose up
  - docker compose up -d
  - docker compose down
  - docker compose start
  - docker compose stop
  - docker compose build
  - docker compose build --no-cache

localhost:7474



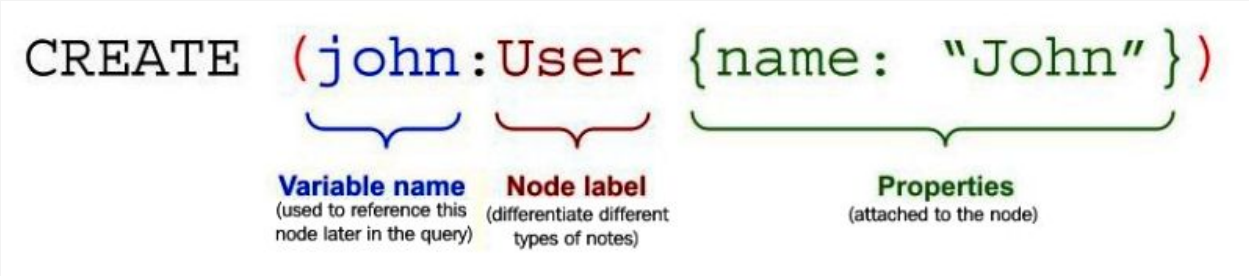
localhost:7474 Then login.

Neo4j Browser



Inserting Data by Creating Nodes

```
CREATE (:User {name: "Alice", birthPlace: "Paris"})
CREATE (:User {name: "Bob", birthPlace: "London"})
CREATE (:User {name: "Carol", birthPlace: "London"})
CREATE (:User {name: "Dave", birthPlace: "London"})
CREATE (:User {name: "Eve", birthPlace: "Rome"})
```



Adding an Edge with No Variable Names

```
CREATE (:User {name: "Alice", birthPlace: "Paris"})
CREATE (:User {name: "Bob", birthPlace: "London"})

MATCH (alice:User {name:”Alice”})
MATCH (bob:User {name: “Bob”})
CREATE (alice)-[:KNOWS {since: “2022-12-01”}]->(bob)
```

Note: Relationships are directed in neo4j.

Matching

Which users were born in London?

```
MATCH (usr:User {birthPlace: “London”})
RETURN usr.name, usr.birthPlace
```

usr.name	usr.birthPlace
"Bob"	"London"
"Carol"	"London"
"Dave"	"London"

Download Dataset and Move to Import Folder

Clone this repo:
https://github.com/PacktPublishing/Graph-Data-Science-with-Neo4j

In **Chapter02/data** of data repo, unzip the **netflix.zip** file

Copy netflix_titles.csv into the following folder where you put your docker compose file

neo4j_db/neo4j_db/import

Importing Data

Basic Data Importing

Type the following into the Cypher Editor in Neo4j Browser

```
LOAD CSV WITH HEADERS
FROM 'file:///netflix_titles.csv' AS line
CREATE(:Movie {
    id: line.show_id,
    title: line.title,
    releaseYear: line.release_year
  }
)
```

Loading CSVs - General Syntax

```
LOAD CSV
[WITH HEADERS]
FROM 'file:///file_in_import_folder.csv'
AS line
[FIELDTERMINATOR ',']
// do stuffs with 'line'
```

Importing with Directors this Time

```
LOAD CSV WITH HEADERS
FROM 'file:///netflix_titles.csv' AS line
WITH split(line.director, ",") as directors_list
UNWIND directors_list AS director_name
CREATE (:Person {name: trim(director_name)})
```

But this generates duplicate Person nodes (a director can direct more than 1 movie)

Importing with Directors Merged

```
MATCH (p:Person) DELETE p


LOAD CSV WITH HEADERS
FROM 'file:///netflix_titles.csv' AS line
WITH split(line.director, ",") as directors_list
UNWIND directors_list AS director_name
MERGE (:Person {name: director_name})
```

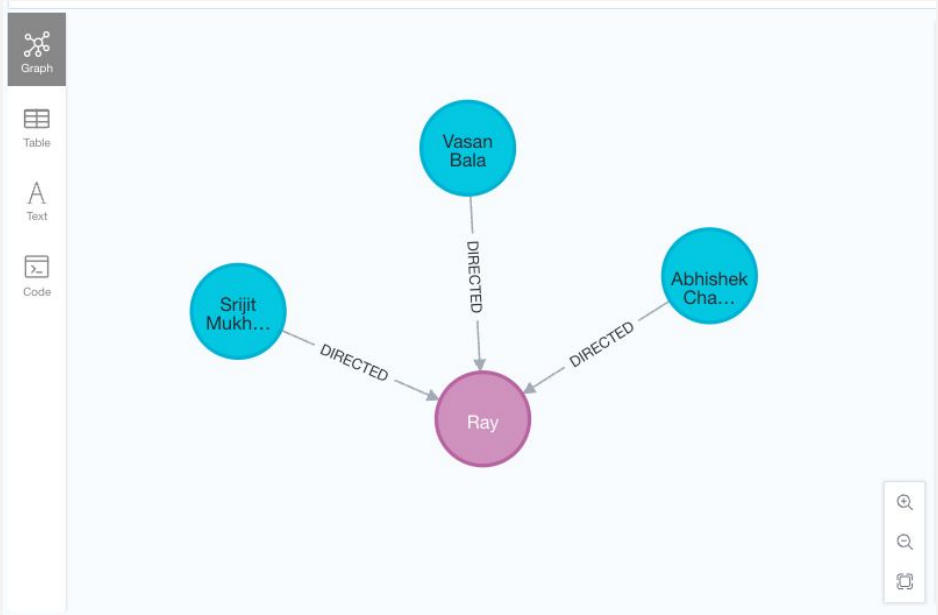
Adding Edges

```
LOAD CSV WITH HEADERS
FROM 'file:///netflix_titles.csv' AS line
MATCH (m:Movie {id: line.show_id})
WITH m, split(line.director, ",") as directors_list
UNWIND directors_list AS director_name
MATCH (p:Person {name: director_name})
CREATE (p)-[:DIRECTED]->(m)
```

Gut Check

Let's check the movie titled Ray:

```
MATCH (m:Movie {title: "Ray"})<-[:DIRECTED]-(p:Person)
RETURN m, p
```



??