1. What is Neo4j?

- A **graph database system** supporting both transactional and analytical graph processing.
- Belongs to a newer class of **NoSQL databases**.
- Features:
    - Schema-optional
    - Various types of indexing
    - ACID-compliant
    - Distributed computing support
- Similar systems: Microsoft CosmosDB, Amazon Neptune

2. Neo4j Query Language & Plugins

- **Cypher**: SQL-like graph query language introduced in 2011
  Example pattern:

  (node)-[:RELATION]->(otherNode)

- **APOC Plugin**: Adds procedures and functions (e.g., string manipulation, path finding)
- **Graph Data Science Plugin**: Offers high-performance implementations of graph algorithms

3. Docker Compose for Neo4j

- **Docker Compose** allows you to define and manage multi-container apps.
- Uses a `docker-compose.yaml` file to define:
    - Services
    - Volumes
    - Networks

4. Example `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
```

Note: Never hard-code secrets in this file. Use a `.env` file.

5. Example `.env` File

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

6. Docker Compose Commands

```
docker --version             # test CLI install
docker compose up              # start containers
docker compose up -d           # detached mode
docker compose down              # stop and remove containers
docker compose start
docker compose stop
docker compose build
docker compose build --no-cache
```

7. Neo4j Browser

- Open in browser at `http://localhost:7474`
- Login using credentials set in `.env`

8. Creating Nodes (Users)

```
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"})
```

9. Creating Relationships (Edges)

```
MATCH (alice:User {name: "Alice"})

MATCH (bob:User {name: "Bob"})

CREATE (alice)-[:KNOWS {since: "2022-12-01"}]->(bob)
```

10. Matching

Find all users born in London:

```
MATCH (usr:User {birthPlace: "London"})

RETURN usr.name, usr.birthPlace
```

11. Importing CSV Data

- Clone: https://github.com/PacktPublishing/Graph-Data-Science-with-Neo4j
- Unzip `netflix.zip` inside `Chapter02/data`
- Move `netflix_titles.csv` to:

  ./neo4j_db/import/

12. Basic Import

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

13. Loading Directors (with Duplicates)

```
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)})
```

14. Loading Directors (with Merge)

```
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})
```

15. Creating Edges (Person → Movie)

```
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)
```

16. Query Example – Directed Movie

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

RETURN m, p
```