

INSTALACIÓN Y CONFIGURACIÓN DE UN ENTORNO DE TRABAJO Neo4j CON GDS



Neo4j

INSTALACIÓN Y CONFIGURACIÓN DE UN ENTORNO DE TRABAJO Neo4J CON GDS

A continuación, se detalla el proceso paso a paso para la instalación de Neo4j y la librería Graph Data Science (GDS) mediante Docker, así como la creación del grafo de trabajo solicitado.

Instalación y Verificación del Entorno

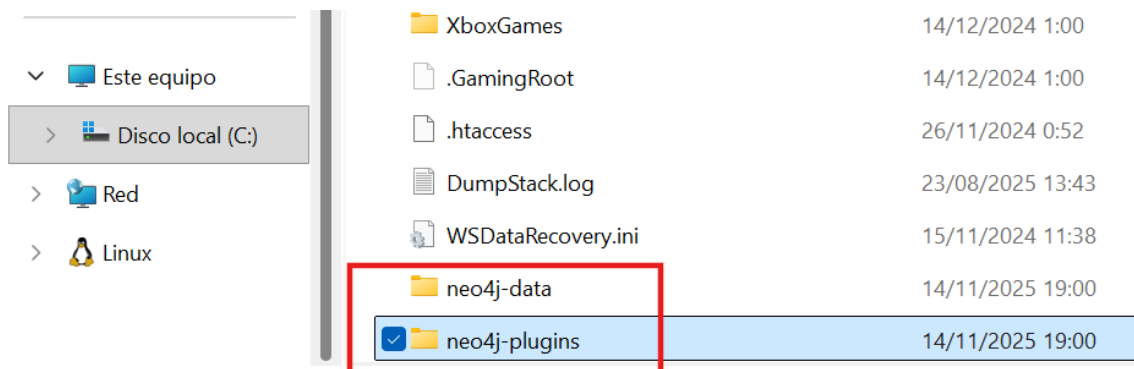
El entorno de trabajo se estableció utilizando Docker para montar una instancia de Neo4j y la librería GDS de manera controlada.

- Preparación de Archivos y Directorios

Se crearon dos carpetas esenciales en el disco local C:\.

C:\neo4j-data: Para la persistencia de los datos de la base de datos

C:\neo4j-plugins: Para la instalación manual de la librería GDS.



Debido a problemas de compatibilidad con la instalación automática, se ha optado por la instalación manual de la librería GDS, utilizando la versión 2.5.4, compatible con Neo4j 4.4.

. El archivo neo4j-graph-data-science-2.5.4.jar se ha descargado desde un repositorio de GitHub y copiado a la carpeta C:\neo4j-plugins

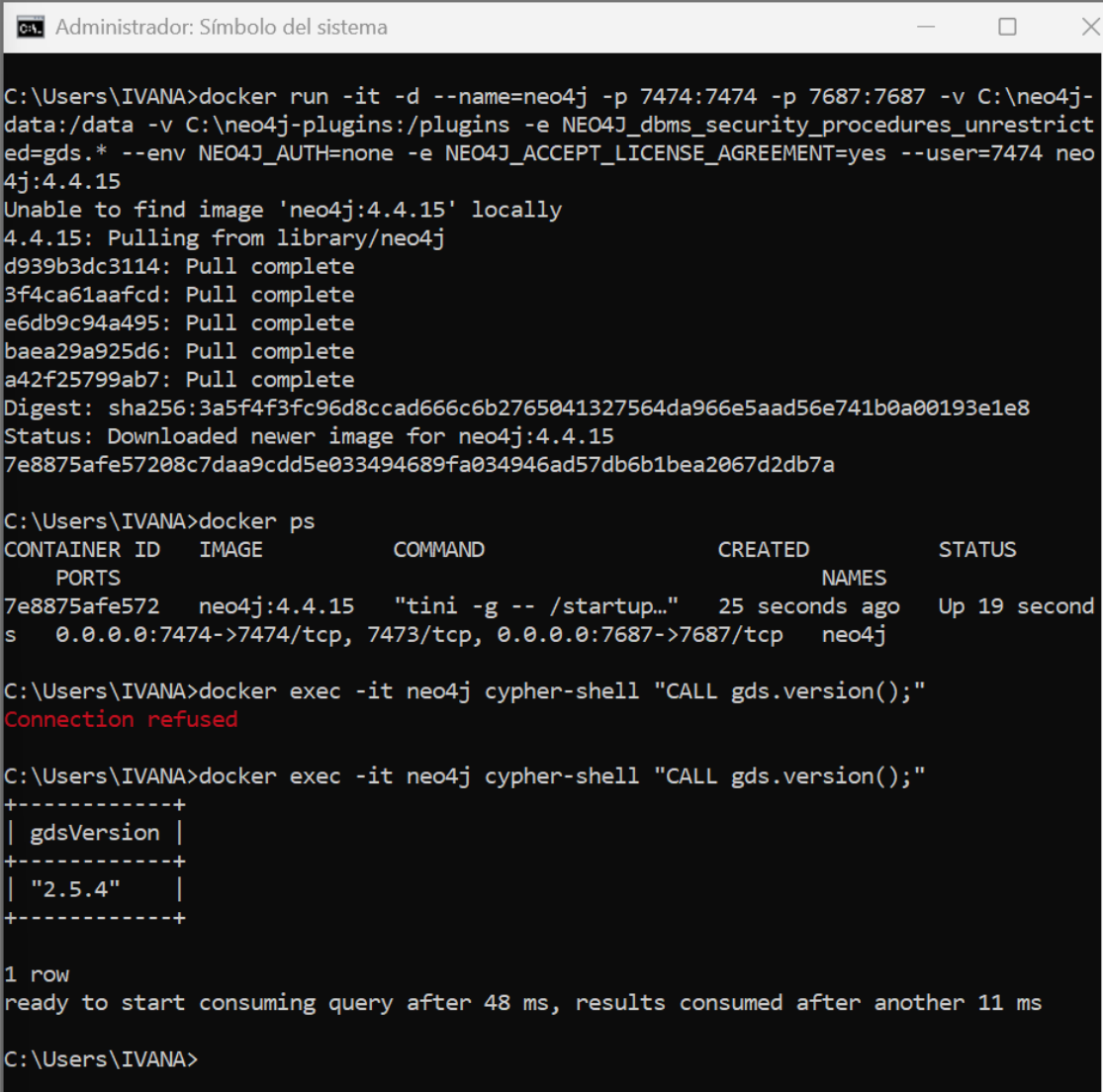
https://github.com/neo4j/graph-data-science/releases?page=3&utm_source=chatgpt.com

- Ejecución del contenedor Docker

Para crear y arrancar el contenedor asegurando la compatibilidad de permisos de volumen (--user=7474) y la carga del plugin, ejecutaremos el siguiente comando:

```
docker run -it -d --name=neo4j -p 7474:7474 -p 7687:7687 -v C:\neo4j-  
data:/data -v C:\neo4j-plugins:/plugins -e  
NEO4J_dbms_security_procedures_unrestricted=gds.* --env  
NEO4J_AUTH=none -e NEO4J_ACCEPT_LICENSE_AGREEMENT=yes --user=7474  
neo4j:4.4.15
```

No pondremos ni usuario ni contraseña.

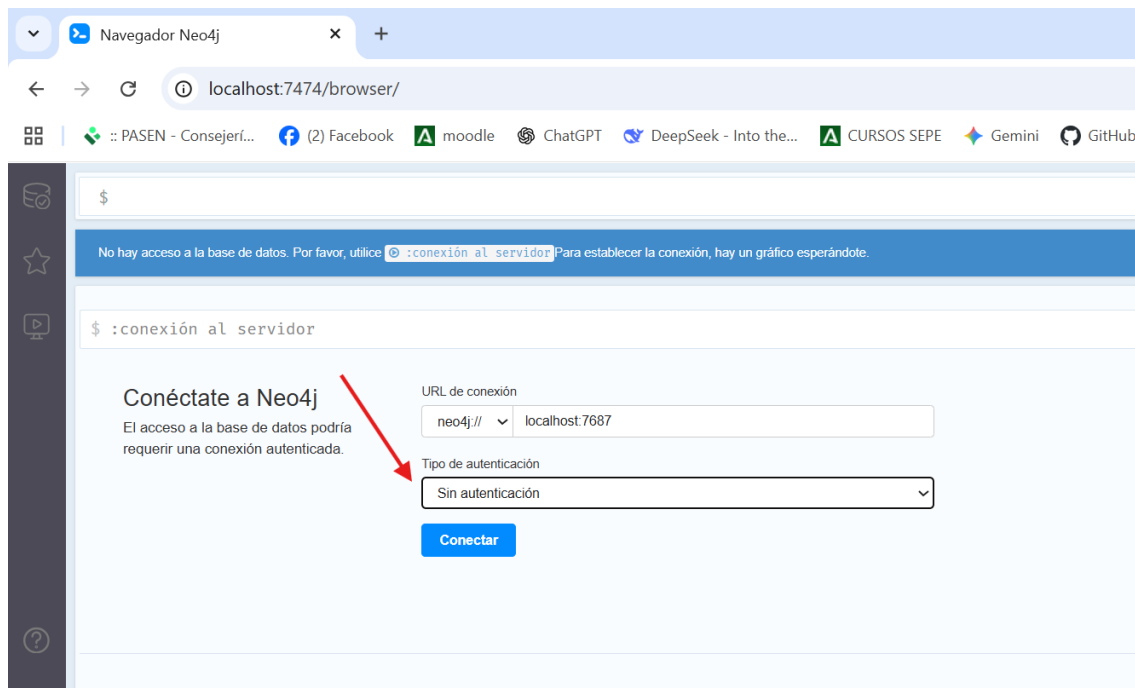


```
C:\Users\IVANA>docker run -it -d --name=neo4j -p 7474:7474 -p 7687:7687 -v C:\neo4j-  
data:/data -v C:\neo4j-plugins:/plugins -e NEO4J_dbms_security_procedures_unrestrict  
ed=gds.* --env NEO4J_AUTH=none -e NEO4J_ACCEPT_LICENSE_AGREEMENT=yes --user=7474 neo  
4j:4.4.15  
Unable to find image 'neo4j:4.4.15' locally  
4.4.15: Pulling from library/neo4j  
d939b3dc3114: Pull complete  
3f4ca61aafcd: Pull complete  
e6db9c94a495: Pull complete  
baea29a925d6: Pull complete  
a42f25799ab7: Pull complete  
Digest: sha256:3a5f4f3fc96d8ccad666c6b2765041327564da966e5aad56e741b0a00193e1e8  
Status: Downloaded newer image for neo4j:4.4.15  
7e8875afe57208c7daa9cdd5e033494689fa034946ad57db6b1bea2067d2db7a  
  
C:\Users\IVANA>docker ps  
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS  
PORTS          NAMES  
7e8875afe572   neo4j:4.4.15  "tini -g -- /startup..." 25 seconds ago Up 19 second  
s  0.0.0.0:7474->7474/tcp, 7473/tcp, 0.0.0.0:7687->7687/tcp  neo4j  
  
C:\Users\IVANA>docker exec -it neo4j cypher-shell "CALL gds.version();"   
Connection refused  
  
C:\Users\IVANA>docker exec -it neo4j cypher-shell "CALL gds.version();"   
+-----+  
| gdsVersion |  
+-----+  
| "2.5.4" |  
+-----+  
  
1 row  
ready to start consuming query after 48 ms, results consumed after another 11 ms  
  
C:\Users\IVANA>
```

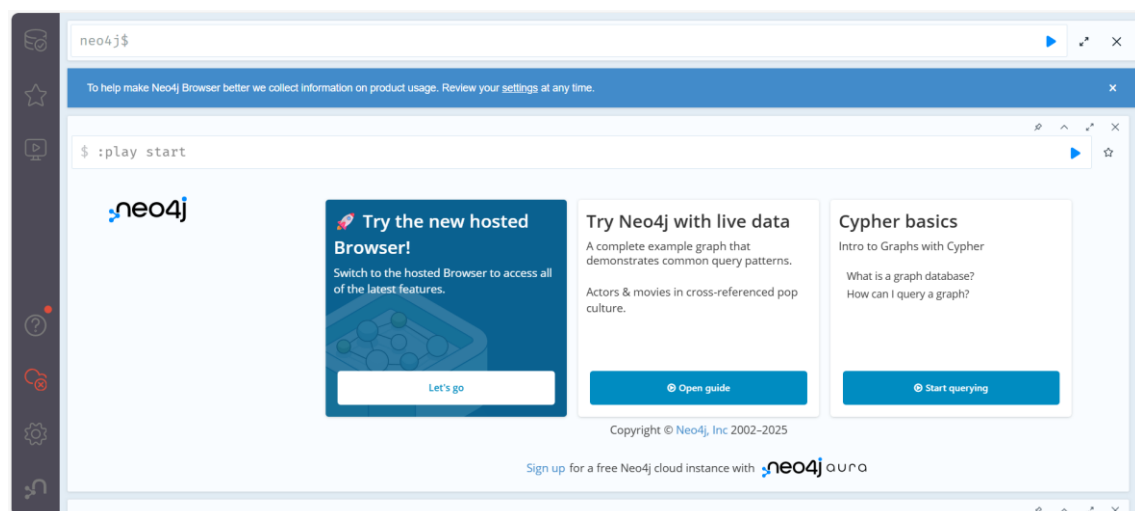
La ejecución confirmó que el contenedor neo4j se inició correctamente y que GDS se instaló perfectamente.

- Verificación de GDS y acceso al navegador.

Una vez que tengamos corriendo el contenedor, esperamos unos 30 segundos y abrimos el navegador en el puerto correspondiente: [//localhost:7687/browser/](http://localhost:7687/browser/) y elegimos la opción→ **sin autenticación** para conectarnos al servidor

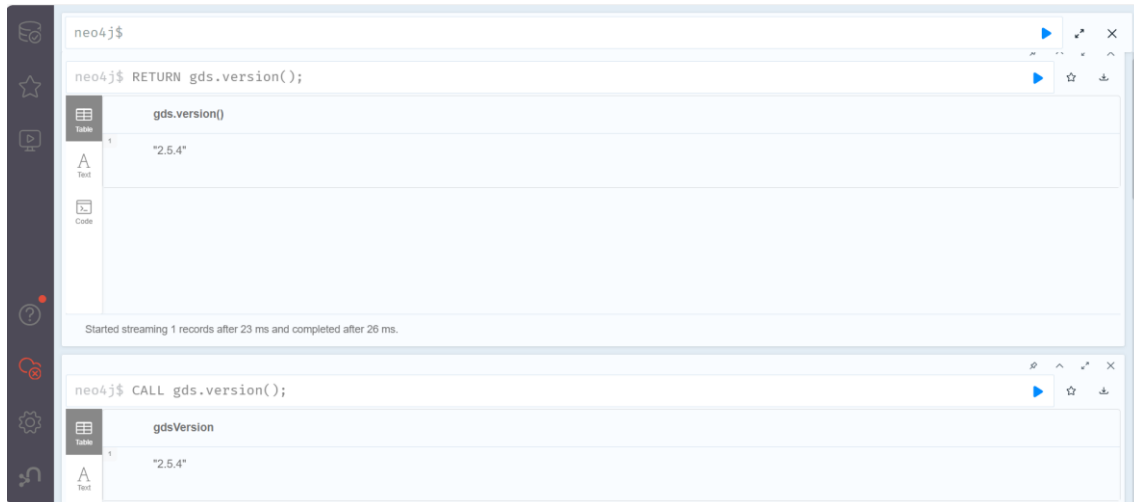


NOTA→ Con Chrome da error de conexión, por lo que es necesario cambiar de navegador. En este caso hemos utilizado Edge.

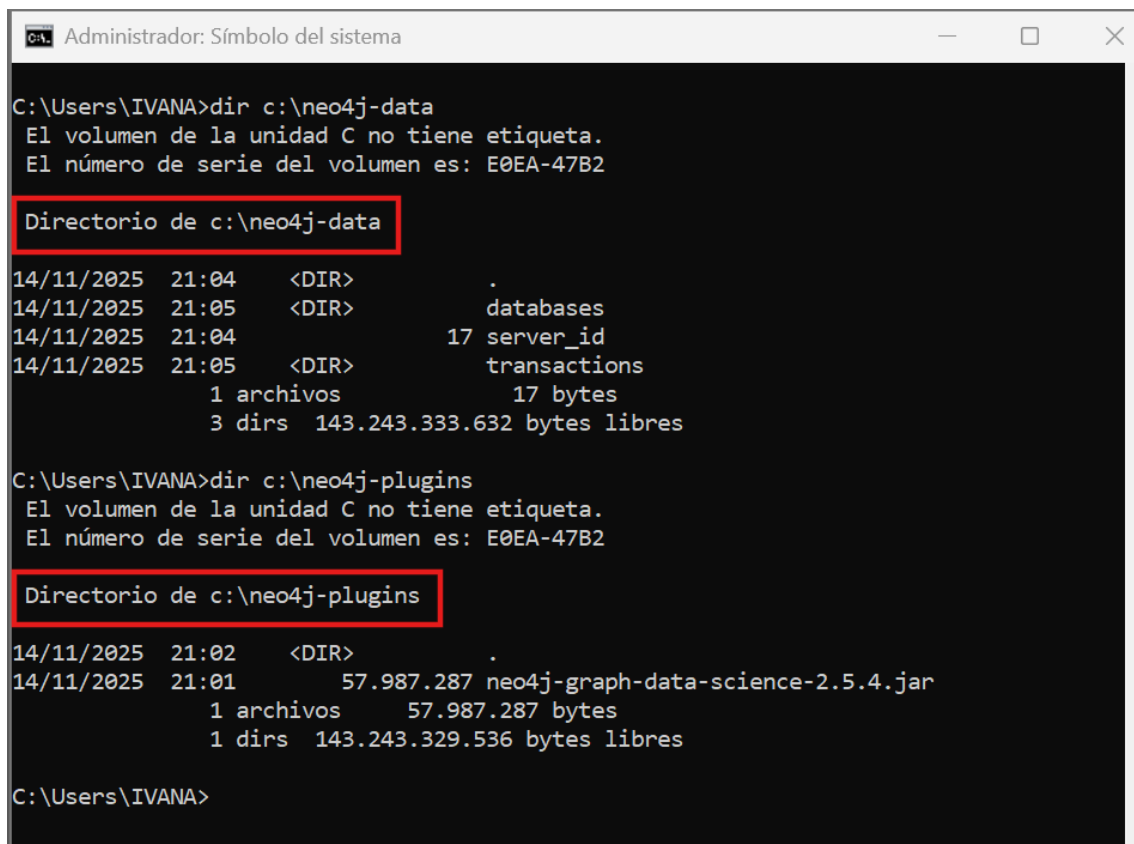


En el terminal de neo4j, se puede verificar igualmente la correcta instalación de GDS. Pueden usarse uno de estos dos comandos:

- `CALL gds.version();`
- `RETURN gds.version();`



Verificamos también el contenido de las carpetas con un simple ***dir*** en el terminal



Creación del Grafo de trabajo

Crea el siguiente grafo en tu entorno de trabajo:



Una vez confirmado el correcto funcionamiento del entorno, procedemos a crear el grafo solicitado, el cual representa una red de ciudades y las distancias entre ellas.

- Estructura del grafo

- Nodos → Usaremos la etiqueta **:City** para representar las ciudades
- Relaciones → Se utilizó la relación **:ROAD** con la propiedad **distance** para almacenar los kilómetros entre las ciudades conectadas

- Comando de creación Cypher

Se creó el grafo mediante un único comando **Cypher**, siguiendo la estructura de nodos y relaciones definidas.

- Los nodos con: **CREATE (ciudad:City {name: 'ciudad'})**
- Las relaciones (carreteras con distancias con: **CREATE (ciudad1)-[:ROAD {distance: <distancia en número>}]>(ciudad2)**

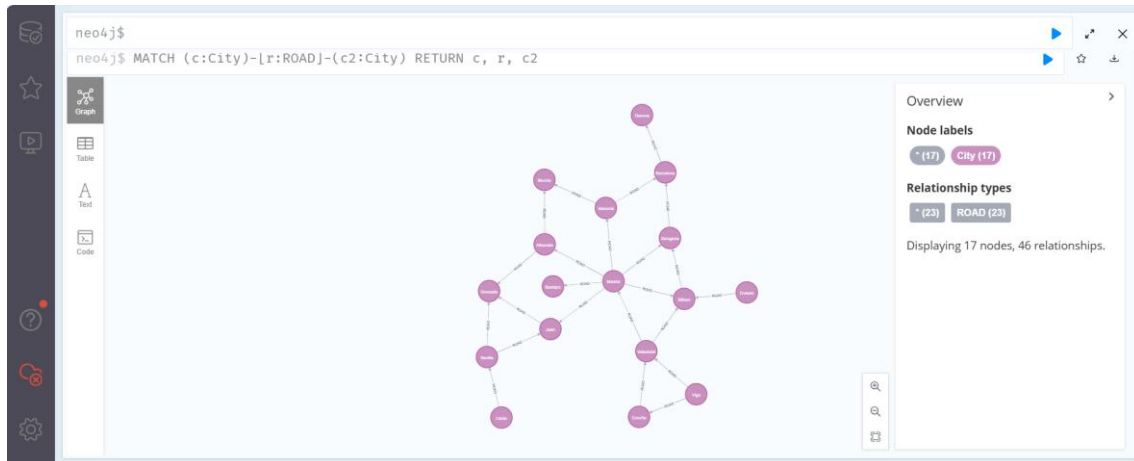
```
neo4j$
15 CREATE (jaen:City {name: 'Jaén'})
16 CREATE (sevilla:City {name: 'Sevilla'})
17 CREATE (cadiz:City {name: 'Cádiz'})
18 CREATE (granada:City {name: 'Granada'})
19
20 // Crea todas las relaciones (carreteras con distancias)
21 CREATE (vigo)-[:ROAD {distance: 171}]->(coruna)
22 CREATE (coruna)-[:ROAD {distance: 455}]->(valladolid)
23 CREATE (vigo)-[:ROAD {distance: 356}]->(valladolid)
24 CREATE (oviedo)-[:ROAD {distance: 304}]->(bilbao)
25 CREATE (valladolid)-[:ROAD {distance: 280}]->(bilbao)
26 CREATE (valladolid)-[:ROAD {distance: 193}]->(madrid)
27
28 // Continúa con las relaciones restantes...
```

Added 17 labels, created 17 nodes, set 40 properties, created 23 relationships, completed after 341 ms.

- Visualización y confirmación

El entorno de trabajo quedó correctamente establecido y, la visualización se confirmó con el comando:

```
MATCH (c:City)-[r:ROAD]-(c2:City)  
RETURN c, r, c2
```



La ejecución mostró 17 nodos y 46 relaciones (debido a la bidireccionalidad por defecto del comando MATCH), confirmando la correcta creación del entorno de trabajo.