Erste Schritte mit dem Neo4j-Python-Treiber

neo4j for python

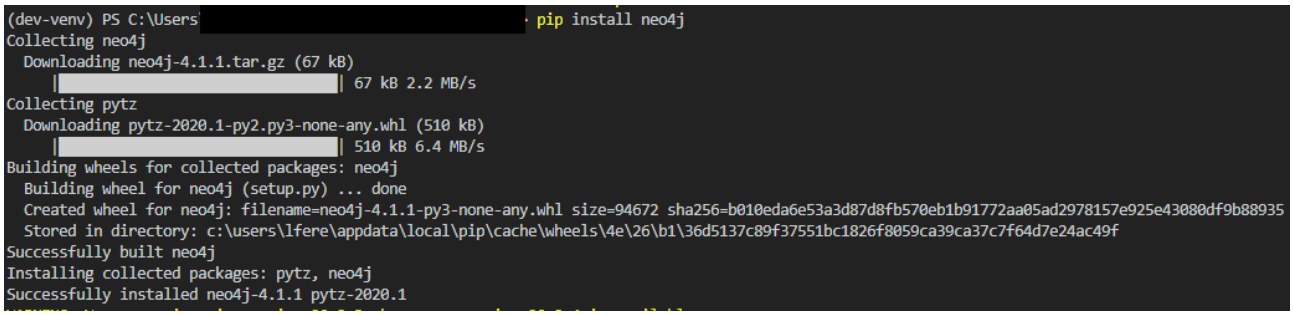
Plattformübergreifend

1. Installieren des offiziellen Python Treibers

Anmerkung: Sollten Sie dem Tutorial aus dem Git-Repository über die ReadMe.md folgen, ist der Neo4j-Treiber bereits in Ihrer Python-Virtuellen-Umgebung installiert (über pip install -r requirements.txt). Ansonsten kann das Paket aber auch manuell über

pip install neo4j

installiert werden.



Das Paket wird offiziell auf der Seite

https://pypi.org/project/neo4j/

bereitgestellt.

2. Verbinden des Treibers mit der Neo4j Instanz

Neo4j-Server-Community muss auf dem System laufen.

Damit Python sich mit der laufenden Instanz verbinden kann, müssen folgende Daten dem Treiber bekannt gegeben werden (ähnlich dem Connection-String einer SQL-Datenbank).

```
# https://neo4j.com/developer/aura-connect-driver/ for Aura specific connection URL
# Connecting to Aura, use the "neo4j+s" URI scheme
  scheme = "neo4j"
  host_name = "localhost"
# Bolt Port https://neo4j.com/docs/operations-manual/current/configuration/ports/ |
# .NET | Java | JavaScript | Go | Python
  port = 7687

  url = f"{scheme}://{host_name}:{port}"
  user = 'neo4j' # your instance-user
  password = 'neo4j'  # your instance-password
```

Der Treiber wird über den folgenden Import geladen und über diesen der Treiber erstellt.

```
from neo4j import GraphDatabase


driver = GraphDatabase.driver(uri, auth=(user, password))
```

3. Ausführen einer Abfrage über den Python Treiber

Eine mögliche Methode zum Ausführen von Cypher-Queries bietet die nachfolgende **__run(tx, query)** Funktion.

```
import logging
from neo4j.exceptions import ServiceUnavailable

def __run(tx, query):
  result = tx.run(query)
   try:
     return [record for record in result]
   except ServiceUnavailable as ex:
     logging.error(f"{query} raised an error: \n {ex}")
     raise
```

Diese Funktion muss entsprechend von einer Treiber-Session aufgerufen werden. Dies geschieht über zum Beispiel den Nachfolgenden Quellcode.

```
query = 'MATCH (n) DETACH DELETE n'

with driver.session() as session:
   # Write transactions allow the driver to handle retries and transient errors
    result = session.write_transaction(
       __run, query
    )
    for record in result:
       print(record)
```

3.1 Häufig keine allgemeine Run-Methode genutzt

Zur Vereinfachung der Python-Anwendung wird häufig keine allgemein-gültige Cypher-Run-Methode wie oben genutzt, sondern selbsgeschriebene Methoden wie **add_friend,** diese haben dann ihren Cypher Code, hardcodiert so vorbereitet, dass nur einzelne eigenschaften an diese Übergeben werden müssen und nicht der gesamte Cypher-String.

Der oben gezeigte Ansatz dient nur zum allgemeinen Verständnis und wird vom Tutorial verwendet, damit die Cypher-String beim Aufruf ausgeschrieben werden müssen.