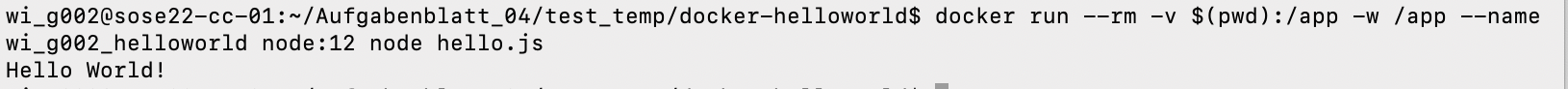
# Dokumentation Aufgabenblatt 4

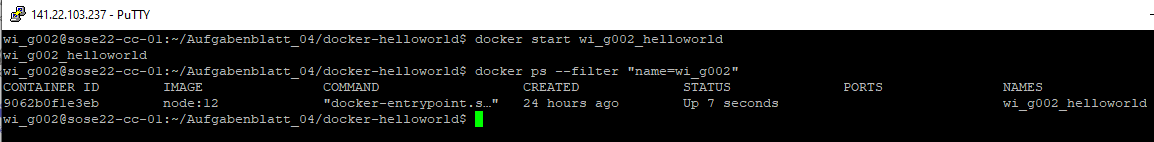
## Aufgabe 1

docker run --rm -v $(pwd):/app -w /app --name wi\_g002\_helloworld node:12 node hello.js

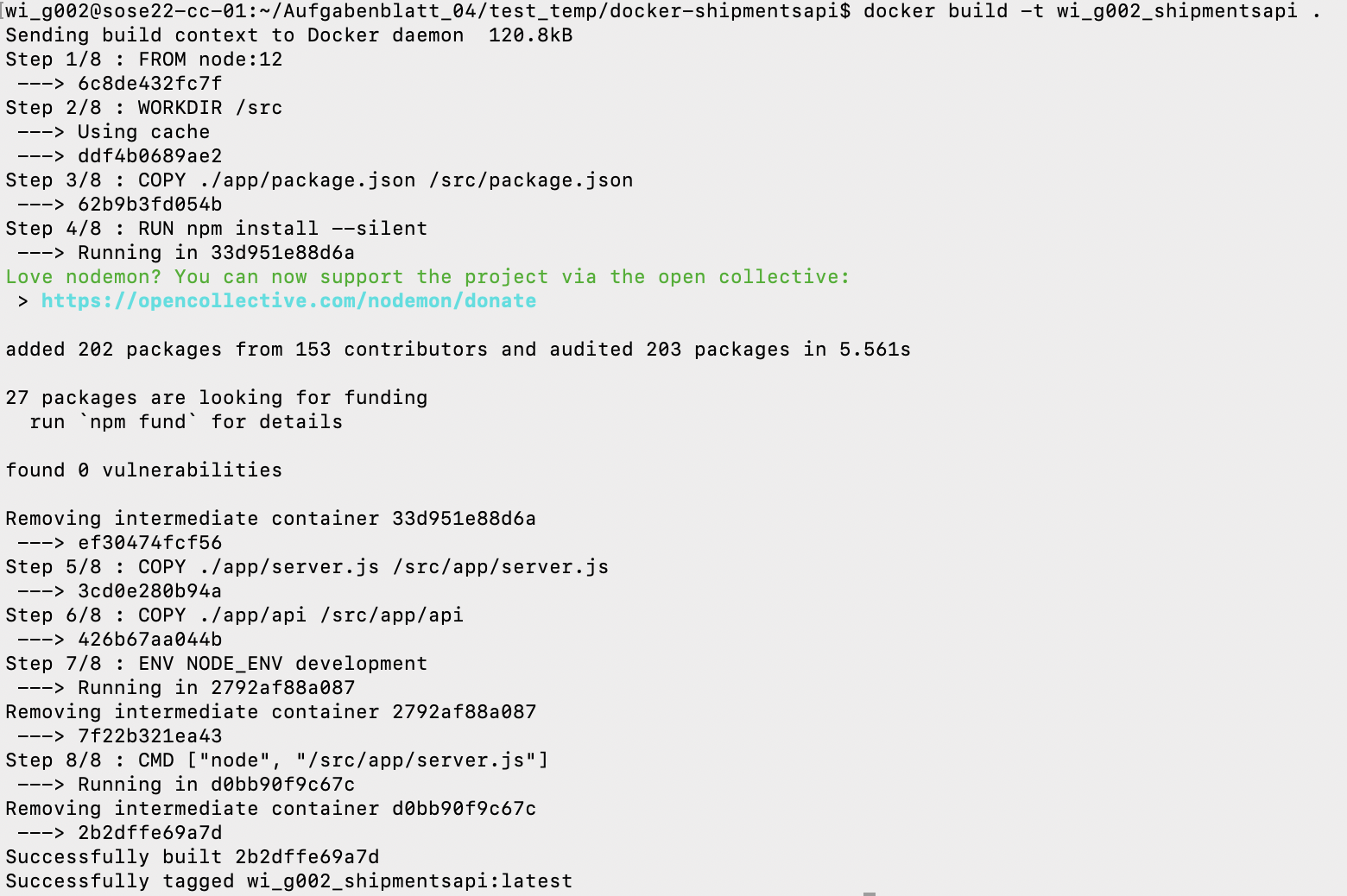


docker run -v $(pwd):/app -w /app --name wi\_g002\_helloworld node:12 node hello.js

Mit docker start wi\_g002\_helloworld können wir nun den Docker auf dem Server starten. Der Test über den “docker ps”-Befehl liefert folgendes Ergebnis:



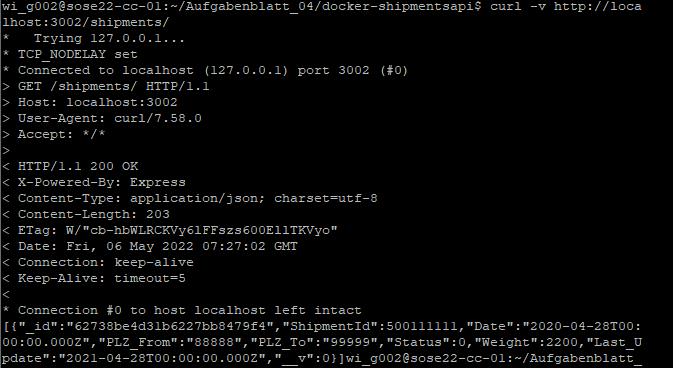
docker build -t wi\_g002\_shipmentsapi



docker run --net=host --name wi\_g002\_\_shipmentsapi wi\_g002\_\_shipmentsapi



curl -v <http://localhost:3002/shipments/>



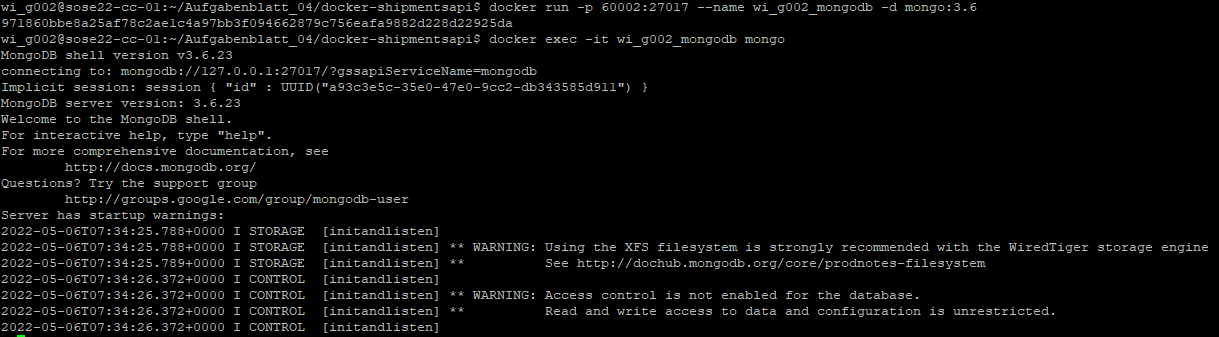
## MongoDB Container

**Anlegen mit**

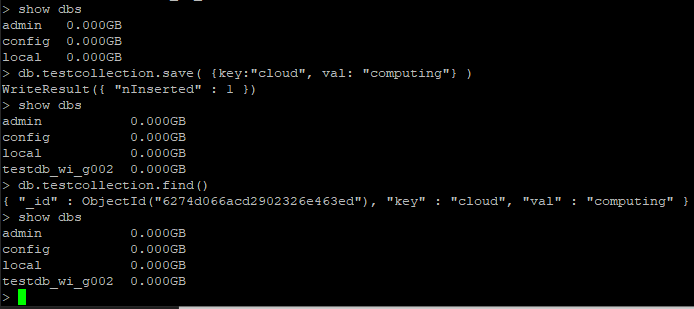
docker run -p 60002:27017 --name wi\_g002\_mongodb -d mongo:3.6

**Und ausführen mit**

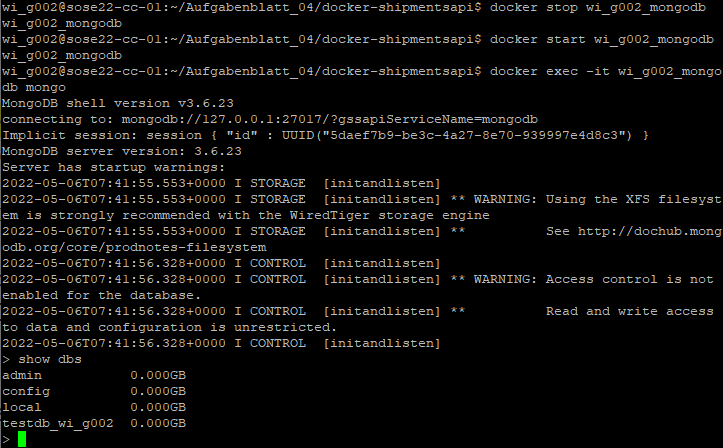
docker exec -it wi\_g002\_mongodb mongo



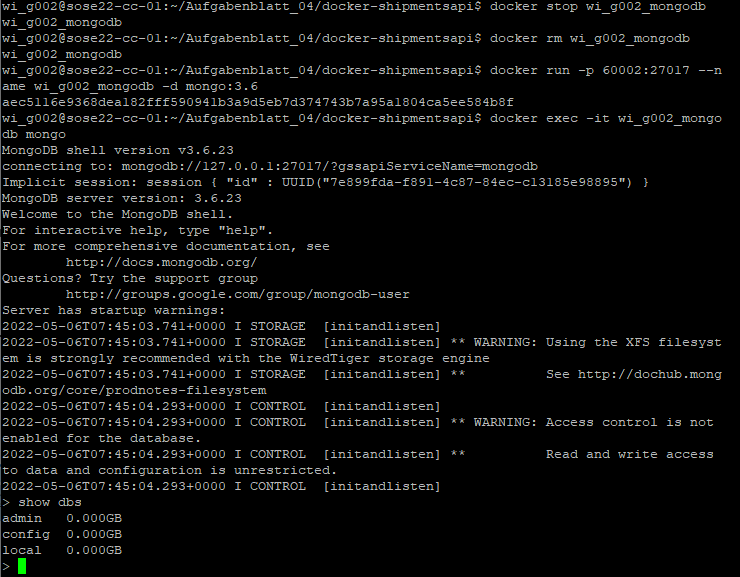
**Datenbank in mongo shell anlegen:**



Neustarten des MongoDB-Dockers & Anzeigen der vorhandenen Datenbanken. Die erstellte testdb\_wi\_g002 ist noch vorhanden.

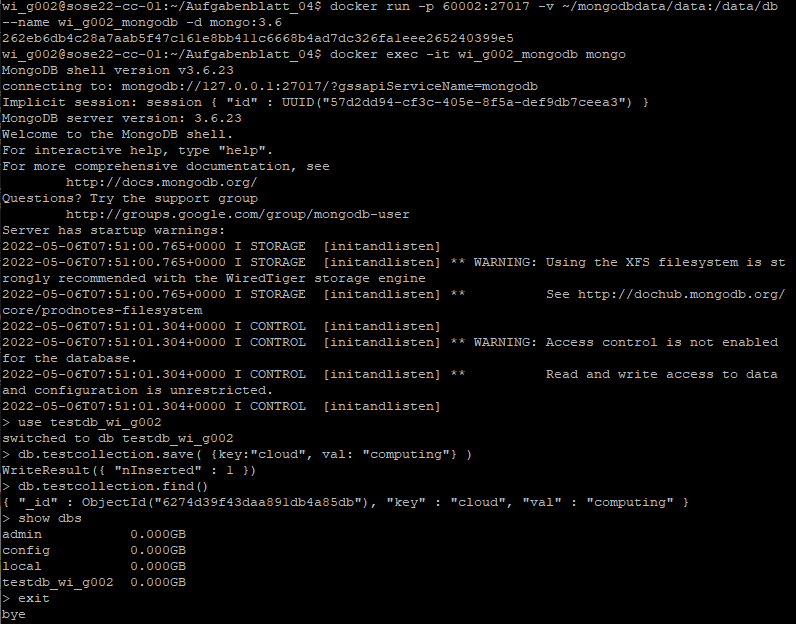


Beenden und entfernen des MongoDB Dockers. Nach einem Neustart ist die erstellte Datenbank nicht mehr vorhanden:

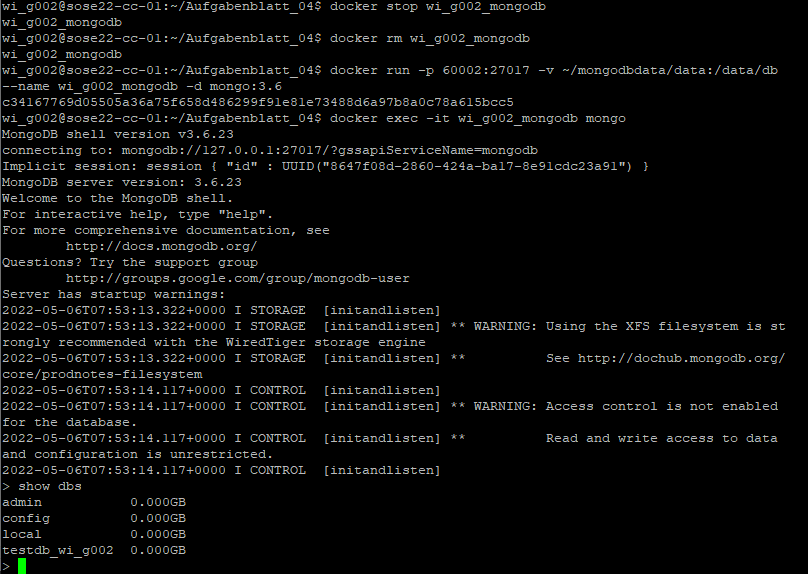


## Persistente Daten

Anlegen des Dockers und der Datenbank mit festem Pfad.



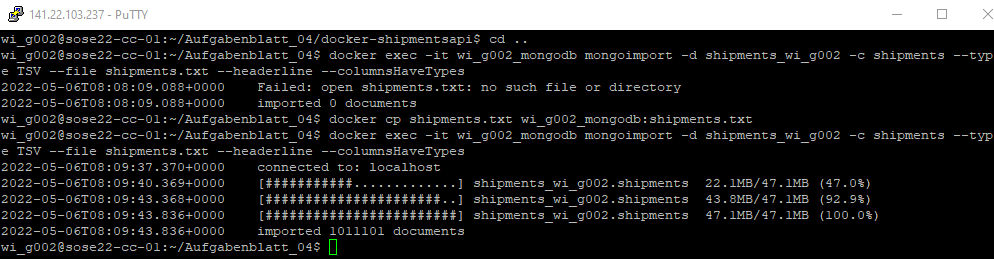
Danach haben wir den Docker geschlossen, entfernt und neu gestartet:



Dieses Mal ist die Testdatenbank noch vorhanden, da sie persistent und nicht im Docker gespeichert wurde.

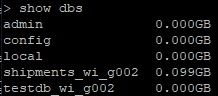
## Daten importieren

Importieren der Datei shipments.txt und vorheriges einfügen in den Container:

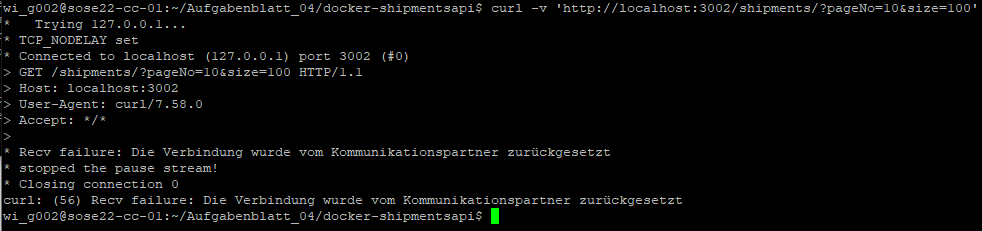


## Verbinden der Datenbank mit der API

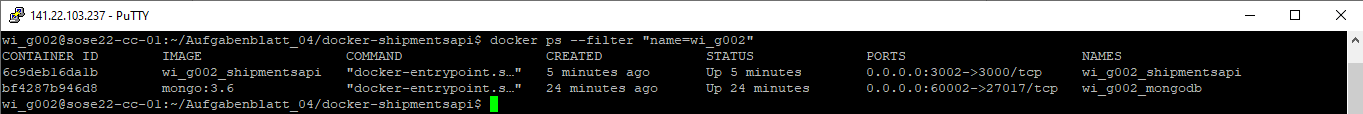
Nach dem Anlegen des Netzwerks und Neustarten des mongoDB-Containers mit genanntem zusätzlichem Parameter ist die importierte Datenbank noch immer vorhanden:



Nach anpassen der Dateien und Neustarten des ShipmentAPI-Dockers, liefert der curl-Befehl:



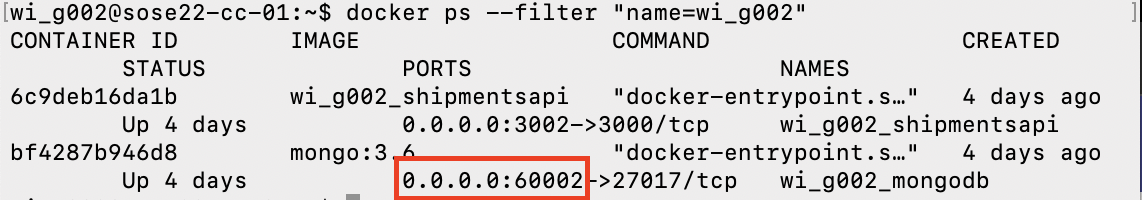
Beide Docker laufen problemlos auf dem Server:



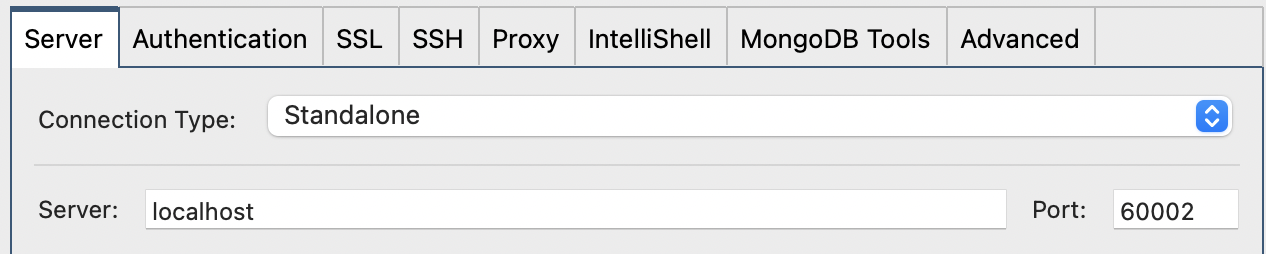
Um auf die Datenbank im Container zuzugreifen, sollte eine neue Verbindung zur Datenbank konfiguriert werden. Die folgende Option wurde verwendet:

* SSH
  + “Use SSH tunnel to connect”
  + SSH adress: 141.22.103.237
  + SSH port: 22
  + Username und Password : Wie beim normalen ssh-Server-Zugang.

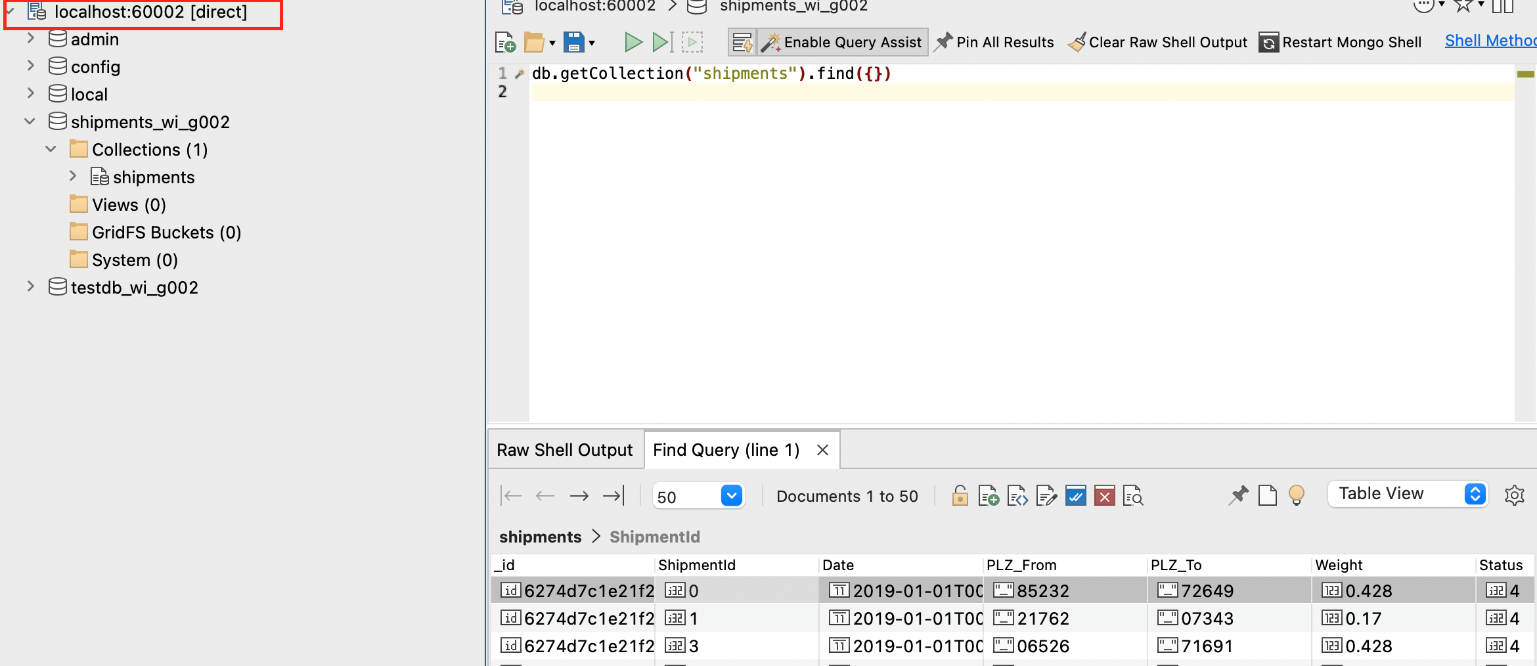
Um die Adresse und den Port für die Verbindung mit dem Container zu ermitteln, verwenden wir den Befehl *docker ps --filter "name=wi\_g002"* und verwenden diese Daten im Fenster Server.



So sieht der Verbindungsbildschirm aus:



Dann können wir die Daten direkt im Container überprüfen.



Nachdem die Container (DB und Shipmentsapi) angepasst wurden, um mit den neuen Funktionen zu kommunizieren, wurden sie fertiggestellt, neu kompiliert und neu gestartet.

Die folgenden Befehle wurden weggelassen, da sie Mongoose-Standardeinstellungen sind und in der vorliegenden Version nicht unterstützt werden.

mongoose.set('useNewUrlParser', true);  
mongoose.set('useFindAndModify', false);  
mongoose.set('useCreateIndex', true);  
mongoose.set('useUnifiedTopology', true);

Mit dem Befehl curl -v '<http://localhost:3002/shipments/?pageNo=10&size=100>' testen wir das Funktionieren der Api.

