Dockerized NoSQL Tutorial

Docker Einführung

Docker bietet eine einfache Möglichkeit, verschiedene Systeme/Programme/Installationen innerhalb einer in sich abgeschlossenen, bereits vorkonfigurierten Systemumgebung komplett isoliert ablaufen zu lassen.
Die aufwändige Installation und Konfiguration von Systemen und Programmen kann dadurch vermieden werden. So genügt z.B. folgender Aufruf, um eine MySQL Datenbank komplett konfiguriert und einsatzbereit zu starten:

```
docker run --name my_mysql -e MYSQL_ROOT_PASSWORD=jenskohler -d -p 3306:3306 mysql
```

Etwas umständlich dabei ist nur die Notwendigkeit, dass u.U. sehr viele Parameter übergeben werden müssen. Wie im obigen Beispiel gezeigt, wird das root Passwort für den Datenbank-Administrator über den Parameter -e gesetzt. Die Bedeutung der weiteren Parameter kann entweder in der Dokumentation des jeweiligen Container-Providers (https://hub.docker.com) oder im Docker Manual (docker --help) nachgelesen werden.
Wichtig ist der Parameter -p (oder --publish): hiermit werden die Ports z.B. 3306, der im Docker-Container läuft auch auf dem Betriebssystem geöffnet. Andernfalls ist kein Zugriff auf den Container möglich.

Die Installation von Docker unterscheidet sich je nach Betriebssystem, ist aber über die Docker-Webseite (https://www.docker.com) gut dokumentiert.

Für die untenstehenden Tests wurde ein Linux Ubuntu 17.04 verwendet. Andere Betriebssysteme unterscheiden sich nur in der Installation von Docker, die jeweiligen Docker-Befehle (z.B. docker run…, docker pull…, etc.) sind überall gleich.

Da es sich bei Docker um eine Art *leichtgewichtige* Virtualisierung handelt, ist zu beachten, dass alle Container nicht zustandsbehaftet (also *stateless*) sind. Daten können nur über sehr umständliche Umwege tatsächlich in Containern gespeichert werden. Nach dem Löschen/Beenden des Containers (z.B. docker stop <container-id>) sind alle Daten innerhalb des Containers weg! Daten können also nicht dauerhaft gespeichert werden. Für Testzwecke allerdings ist dies nicht sonderlich relevant, da die Container hier einfach bei jedem Start neu mit Testdaten gefüllt werden können.

Nützliche Kommandos (unter Linux):

- sudo netstat -tuplen
  Prüfen, ob ein Container seine Ports wirklich auf dem Betriebssystem geöffnet hat:
- docker ps
  Prüfen, welche Container bereits laufen
- docker stop <container-id>
  Einen laufenden Container beenden
- docker rm <container-id>
  Einen Container löschen

Beispiel MySQL:
docker run --name my_mysql -e MYSQL_ROOT_PASSWORD=jenskohler -d -p 3306:3306 mysql

NoSQL

Folgende (erfolgreich getestete) NoSQL Datenbanken können die CRUD-Operationen bewerkstelligen:

MongoDB:
mkdir ~/data
docker run --name my_mongodb -d -p 27017:27017 -v ~/data:/data/db mongo

Neo4J:
docker run -d --publish=7474:7474 --publish=7687:7687 --volume=$HOME/neo4j/data:/data --volume=$HOME/neo4j/logs:/logs neo4j:3.0

Danach kann mit http://localhost:7474 auf die neo4j Weboberfläche zugegriffen werden:
Admin Username: neo4j
Password: neo4j
Dieses muss sofort nach dem ersten Login (s.o.) geändert werden.

Redis:
docker run --name my_redis -d -p 6379:6379 redis

Infinispan:
docker run -d -it -p 8080:8080 -p 11222:11222 --rm --name=my_infinispan jboss/infinispan-server

Cassandra:
docker run --name my_cassandra -d -p 9042:9042 cassandra