**Hands-On-Session: Erstellung eines lokalen MySQL NDB Clusters mit Docker**

Es soll ein Cluster-Netzwerk mit folgender Konfiguration erstellt werden:

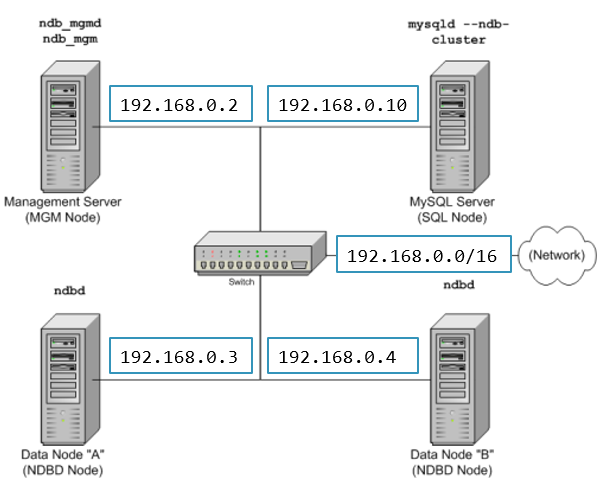


* Ein Management-Knoten ndb\_mgmd
* Ein MySQL-Server-Knoten mysqld
* Zwei Datenknoten ndbd

**Voraussetzungen:**

1. Installation einer Docker-Umgebung: „Docker-Desktop“ (<https://www.docker.com/get-started>)
2. Pull des MySQL NDB-Cluster-Images über die Kommandozeile:

docker pull mysql/mysql-cluster



**Hands-On-Session 1: Konfiguration der Cluster-Datenbank**

1. Anlage eines lokalen Cluster-Netzwerks im lokalen IP-Raum

docker network create cluster --subnet=192.168.0.0/16

1. „Management node“ ndb\_mgmd anlegen

docker run -d --net=cluster --name=management1 --ip=192.168.0.2 -p 0.0.0.0:3300:3300 mysql/mysql-cluster ndb\_mgmd

1. „Data nodes“ ndbd anlegen
2. Datenknoten

docker run -d --net=cluster --name=ndb1 --ip=192.168.0.3 -p 0.0.0.0:3301:3301 mysql/mysql-cluster ndbd

1. Datenknoten

docker run -d --net=cluster --name=ndb2 --ip=192.168.0.4 -p 0.0.0.0:3302:3302 mysql/mysql-cluster ndbd

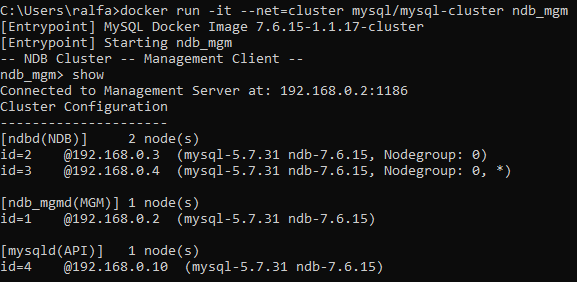
1. „MySQL Server node“ (API) anlegen

docker run -d --net=cluster --name=mysql\_1 --ip=192.168.0.10 -p 0.0.0.0:3306:3306 -e MYSQL\_GENERATE\_ROOT\_PASSWORD=true mysql/mysql-cluster mysqld -–skip-name-resolve

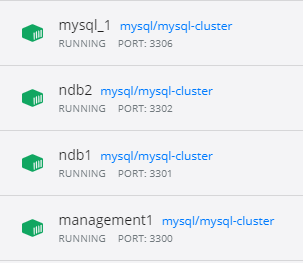
1. „Management Client“ ndb\_mgm anlegen, welcher die Verbindung mit dem Management-Server aufnimmt.

docker run -it --net=cluster mysql/mysql-cluster ndb\_mgm

Der Befehl „show“ gibt einen Überblick über die angelegten Knoten des Clusters:



Im Docker Desktop werden ebenfalls 4 Container-Instanzen angelegt:



1. Für den Aufbau einer Datenbankverbindung über eine Applikation (Übungsaufgabe) müssen noch die Berechtigungen angepasst werden, zum Beispiel für User ‚root‘:
   * Verbindung zur MySQL-Server-Node aufbauen:

docker exec -it mysql\_1 mysql -uroot -p

* + Default-Passwort aus dem Log des Containers im Docker Desktop entnehmen
  + Neues Passwort setzen:

ALTER USER 'root'@'localhost' IDENTIFIED BY '';

* + Berechtigungen vergeben:

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'localhost';

FLUSH PRIVILEGES;

* + Zugriff von jedem Host zulassen:

USE mysql;

UPDATE user SET host = '%' WHERE user = 'root';

* + MySQL-Server Node neustarten: docker restart mysql\_1

1. Anlage einer Datenbank und Tabellen für die Übungsaufgabe über die Kommandozeile:
   * CREATE DATABASE kfru;

Tabelle: students

CREATE TABLE IF NOT EXISTS `kfru`.`students` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`firstname` VARCHAR(30) NULL DEFAULT NULL,

`lastname` VARCHAR(30) NULL DEFAULT NULL,

`email` VARCHAR(50) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = NDBCLUSTER

DEFAULT CHARACTER SET = latin1;

Tabelle: grading

CREATE TABLE IF NOT EXISTS `kfru`.`grading` (

`students\_id` INT(11) NOT NULL,

`subject` VARCHAR(45) NOT NULL,

`grade` INT(11) NOT NULL,

`passed` VARCHAR(1) NULL DEFAULT NULL,

`submission\_date` DATE NULL DEFAULT NULL,

PRIMARY KEY (`students\_id`, `subject`, `grade`),

CONSTRAINT `fk\_grading\_students`

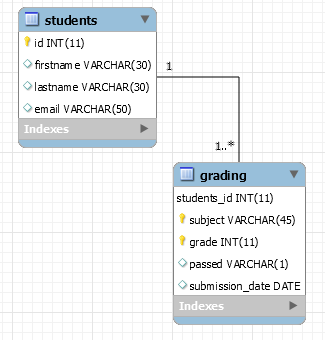
FOREIGN KEY (`students\_id`)

REFERENCES `kfru`.`students` (`id`))

ENGINE = NDBCLUSTER

DEFAULT CHARACTER SET = latin1;

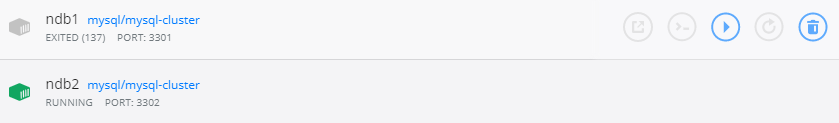
Datenmodellierung:



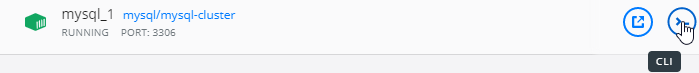
**Hands-On Session 2: Simulation der Hochverfügbarkeit: Ausfall eines Datenknotens**

1. Stoppe einen der beiden Datenknoten (ndb1 oder ndb2) über den Docker Desktop oder die Kommandozeile:

docker stop ndb1



1. Füge einen neuen Datensatz in die Datenbank ein, z.B. über die Kommandozeile bzw. Docker CLI

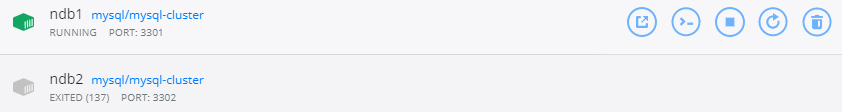


Die Verbindung zur Datenbank erfolgt über den folgenden Befehl:

Mysql -uroot -p

CRUD-Operationen werden über SQL getätigt.

1. Starte nun den gestoppten Datenknoten wieder und stoppe anschließend den zweiten Datenknoten.



1. Prüfe ob der angelegte Datensatz vorhanden ist. Aufgrund der automatischen Bildung von Repliken sollte der neue Datensatz zu finden ein.