

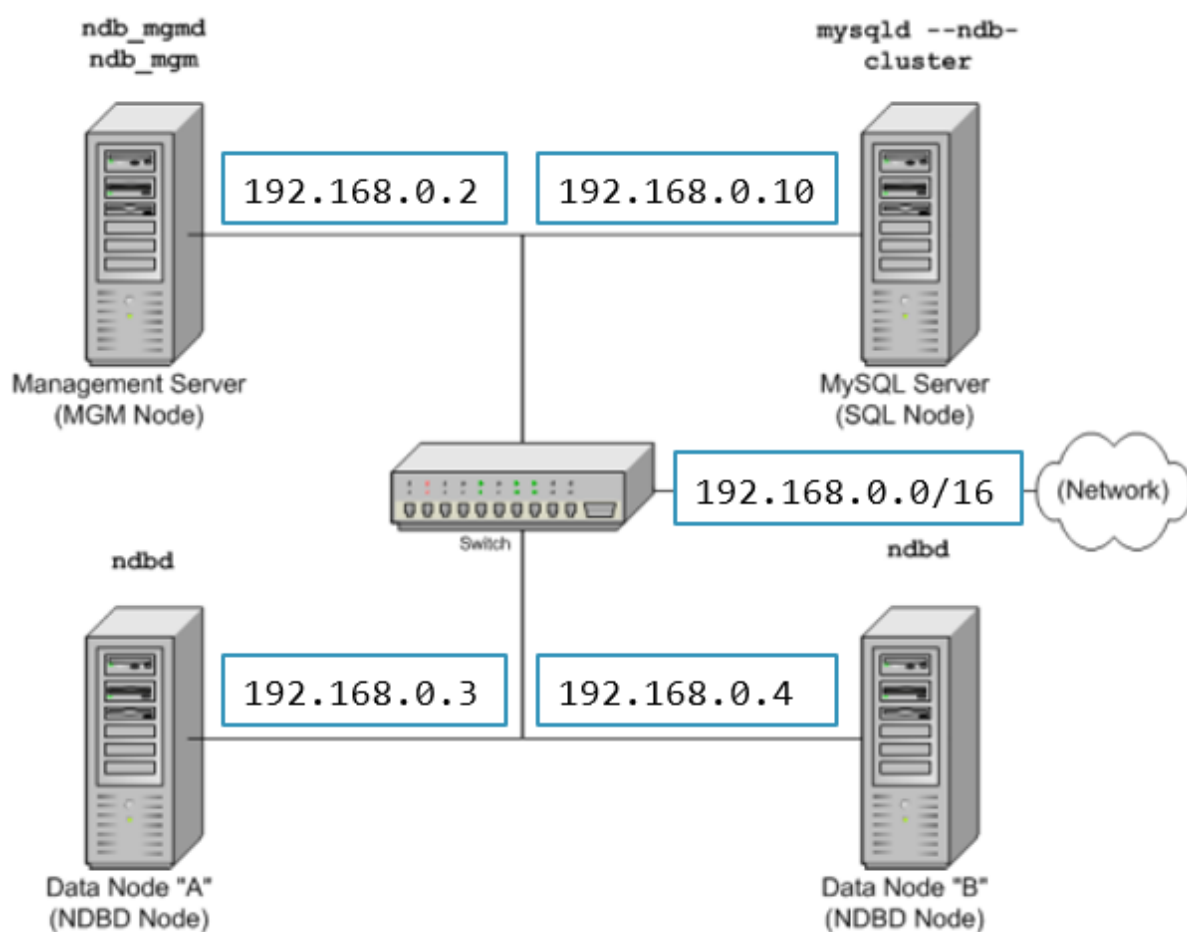
Hands-On-Session: Erstellung eines lokalen MySQL NDB Clusters mit Docker

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. Anlage eines lokalen Cluster-Netzwerks
`docker network create cluster --subnet=192.168.0.0/16`
2. „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`
3. „Data nodes“ `ndbd` anlegen
 1. Datenknoten

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

2. Datenknoten

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

4. „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
```





5. „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:

```
C:\Users\ralfa>docker run -it --net=cluster mysql/mysql-cluster ndb_mgm
[Entrypoint] MySQL Docker Image 7.6.15-1.1.17-cluster
[Entrypoint] Starting ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: 192.168.0.2:1186
Cluster Configuration
-----
[ndbd(NDB)] 2 node(s)
id=2 @192.168.0.3 (mysql-5.7.31 ndb-7.6.15, Nodegroup: 0)
id=3 @192.168.0.4 (mysql-5.7.31 ndb-7.6.15, Nodegroup: 0, *)
[ndb_mgmd(MGM)] 1 node(s)
id=1 @192.168.0.2 (mysql-5.7.31 ndb-7.6.15)
[mysqld(API)] 1 node(s)
id=4 @192.168.0.10 (mysql-5.7.31 ndb-7.6.15)
```

Im Docker Desktop werden ebenfalls 4 Container-Instanzen angelegt:

	mysql_1	mysql/mysql-cluster	RUNNING	PORT: 3306
	ndb2	mysql/mysql-cluster	RUNNING	PORT: 3302
	ndb1	mysql/mysql-cluster	RUNNING	PORT: 3301
	management1	mysql/mysql-cluster	RUNNING	PORT: 3300

6. Für den Aufbau einer Datenbankverbindung über eine Applikation 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`

7. Anlage einer Datenbank und Tabellen für die Übungsaufgabe

- `CREATE DATABASE kfpu;`

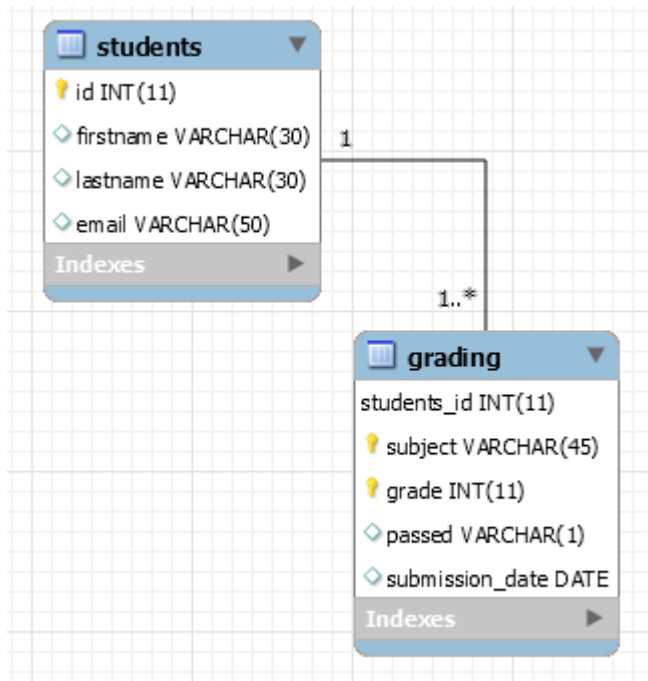
Tabelle: students

```
CREATE TABLE IF NOT EXISTS `kfpu`.`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

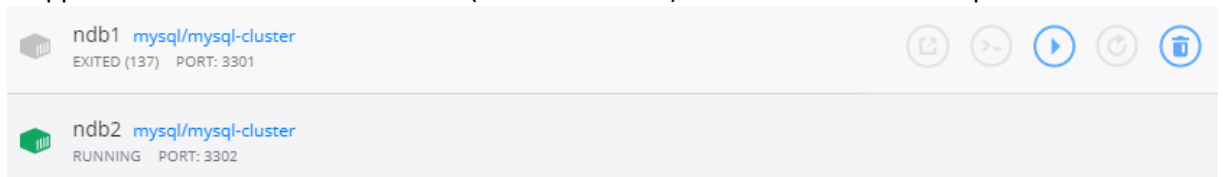
```
CREATE TABLE IF NOT EXISTS `kfpu`.`grading` (
  `students_id` INT(11) NOT NULL,
  `subject` VARCHAR(45) NOT NULL,
  `grade` INT(11) NOT NULL,
  `passed` VARCHAR(1) NULL DEFAULT NULL,
  `submission_date` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`students_id`, `subject`, `grade`),
  CONSTRAINT `fk_grading_students`
    FOREIGN KEY (`students_id`)
    REFERENCES `kfpu`.`students` (`id`))
ENGINE = NDBCLUSTER
DEFAULT CHARACTER SET = latin1;
```

Datenmodellierung:



Simulation der Hochverfügbarkeit: Ausfall eines Datenknotens

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



2. Füge einen neuen Datensatz in die Datenbank ein.
3. Starte den gestoppten Datenknoten wieder und stoppe anschließend den anderen Datenknoten.



4. Prüfe ob der angelegte Datensatz vorhanden ist.