F1F

Ich kann für eine NoSQL Datenbank eine Replikation anwenden.

Lernziele

- Replikation in NoSQL einer Datenbanken verstehen.
- Replikation in NoSQL einer Datenbanken anwenden.
- Replikation in NoSQL einer Datenbanken testen.

Umsetzung

Replikation verstehen

Replikation ist der Prozess, bei dem Daten von einem Knoten (Master) auf einen oder mehrere andere Knoten (Replikate) kopiert werden. Dies geschieht in der Regel in Echtzeit oder nahezu in Echtzeit, um sicherzustellen, dass alle Knoten über die aktuellsten Daten verfügen.

Replikation anwenden mit MongoDB

1. **Erstellen eines Replica Sets**: Ein Replica Set ist eine Gruppe von MongoDB-Instanzen, die dieselben Daten enthalten. Um ein Replica Set zu erstellen, müssen mindestens drei Knoten vorhanden sein. Repilca Set ist eine Primary and Replica replication, das heisst, dass ein Knoten als primär fungiert und die anderen Knoten als Replikate fungieren.
2. **Konfigurieren der Knoten**: Jeder Knoten im Replica Set muss konfiguriert werden, um sicherzustellen, dass er als Replikat oder primärer Knoten fungiert.
3. **Starten des Replica Sets**:

   Um das Replica Set zu starten, den Befehl `rs.initiate()` in der MongoDB-Shell verwenden.
4. **Überprüfen des Replica Sets**:

   Den Befehl `rs.status()` verwenden, um den Status des Replica Sets zu überprüfen und sicherzustellen, dass alle Knoten korrekt verbunden sind.
5. **Testen der Replikation**: Daten in den primären Knoten einfügen und überprüfen, ob sie auf den Replikaten verfügbar sind.

Umsetzung

Docker-Container für MongoDB Replica Set erstellen

```
services:
```

```
  mongo1:
    image: mongo
    container_name: mongo1
    ports:
      - "27017:27017"
    command: ["mongod", "--replSet", "rs0"]

  mongo2:
    image: mongo
    container_name: mongo2
    ports:
      - "27018:27017"
    command: ["mongod", "--replSet", "rs0"]

  mongo3:
    image: mongo
    container_name: mongo3
    ports:
      - "27019:27017"
    command: ["mongod", "--replSet", "rs0"]
```

Verbindung zu mongo1

```
docker exec -it mongo1 mongosh
```

Replica Set initialisieren

```
rs.initiate({
  _id: "rs0",
  members: [
    { _id: 0, host: "mongo1:27017" },
    { _id: 1, host: "mongo2:27017" },
    { _id: 2, host: "mongo3:27017" }
  ]
})
```

Replica Set überprüfen

```
rs.status()
```

Master Knoten bestätigen

```
rs.isMaster().ismaster
```

Daten in den primären Knoten einfügen

Daten in den primären Knoten einfügen

```
use testdb
db.test.insertOne({ message: "replica set test" })
```

Daten auf dem Replikat überprüfen

```
docker exec -it mongo2 mongosh

db.getMongo().setReadPref("secondary")

use testdb
db.test.find()
```

Erwartetes Ergebnis

```
{ _id: ObjectId(...), message: "replica set test" }
```

Nachweis

Replica Set initialisieren

```
test> rs.initiate({
...   _id: "rs0",
...   members: [
...     { _id: 0, host: "mongo1:27017" },
...     { _id: 1, host: "mongo2:27017" },
...     { _id: 2, host: "mongo3:27017" }
...   ]
... })
...
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1747915967, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1747915967, i: 1 })
}
rs0 [direct: secondary] test> rs.status()
{
  set: 'rs0',
  date: ISODate('2025-05-22T12:13:10.941Z'),
```

Status Check

```
rs0 [direct: secondary] test> rs.status()
{
  set: 'rs0',
  date: ISODate('2025-05-22T12:13:10.941Z'),
```

```
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1747915978, i: 16 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2025-05-22T12:12:58.997Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1747915978, i: 16 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1747915978, i: 16 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1747915978, i: 16 }), t: Long('1') },
    writtenOpTime: { ts: Timestamp({ t: 1747915978, i: 16 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2025-05-22T12:12:58.997Z'),
    lastDurableWallTime: ISODate('2025-05-22T12:12:58.997Z'),
    lastWrittenWallTime: ISODate('2025-05-22T12:12:58.997Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1747915967, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2025-05-22T12:12:58.737Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1747915967, i: 1 }), t: Long('-1'
```

Daten in den primären Knoten einfügen

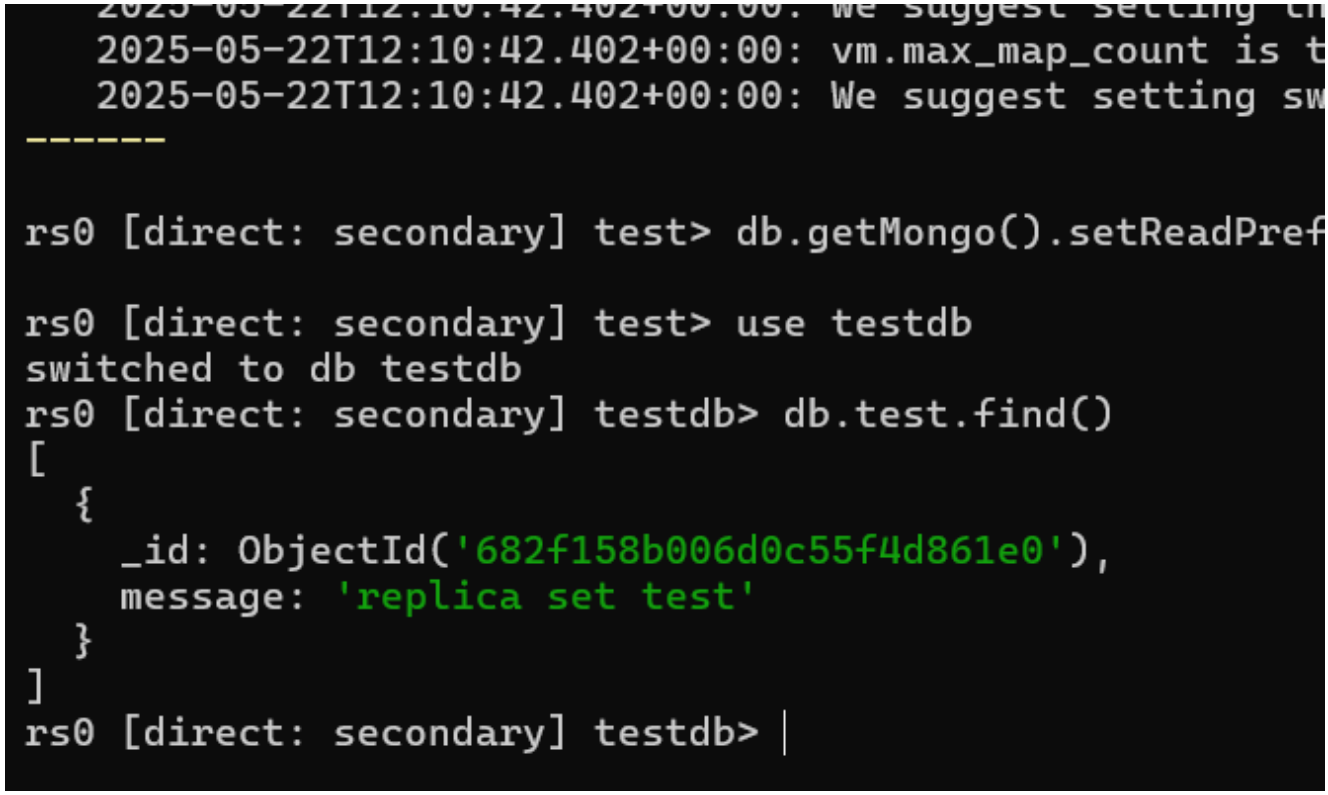
```
rs0 [direct: primary] test> rs.isMaster().ismaster
true
rs0 [direct: primary] test> use testdb
switched to db testdb
rs0 [direct: primary] testdb> db.test.insertOne({ message: "replica set test" })
{
  acknowledged: true,
  insertedId: ObjectId('682f158b006d0c55f4d861e0')
}
```

Daten auf dem Replikat überprüfen

```
C:\Users\user1>docker exec -it mongo2 mongosh
Current Mongosh Log ID: 682f159f64b8df0a74d861df
Connecting to:          mongodb://127.0.0.1:27017/?dire
Using MongoDB:          8.0.9
Using Mongosh:          2.5.0

For mongosh info see: https://www.mongodb.com/docs/mong

------
   The server generated these startup warnings when boo
   2025-05-22T12:10:41.261+00:00: Using the XFS filesys
odnotes-filesystem
   2025-05-22T12:10:42.402+00:00: Access control is not
   2025-05-22T12:10:42.402+00:00: For customers running
```


2025-05-22T12:10:42.402+00:00: vm.max_map_count is t
2025-05-22T12:10:42.402+00:00: We suggest setting sw
rs0 [direct: secondary] test> db.getMongo().setReadPref
rs0 [direct: secondary] test> use testdb
switched to db testdb
rs0 [direct: secondary] testdb> db.test.find()
[
{
_id: ObjectId('682f158b006d0c55f4d861e0'),
message: 'replica set test'
}
]
rs0 [direct: secondary] testdb>