```
*** Database Replication ***
```

Create a Docker network
 docker network create mongo-cluster

```
PS C:\Users\razva> docker network create mongo-cluster
4cd4fa43b8c83d4bf54baf045ab214ec634e5f23834a61db829128a189848a70
PS C:\Users\razva>
```

2. Create three independent MongoDB nodes

docker run -d --name mongo-node1 -p 27800:27017 --net mongo-cluster mongo --replSet replicaset\_test

docker run -d --name mongo-node2 -p 27801:27017 --net mongo-cluster mongo --replSet replicaset test

docker run -d --name mongo-node3 -p 27802:27017 --net mongo-cluster mongo --replSet replicaset test

```
PS C:\Users\razva> docker run -d --name mongo-nodel -p 27800:27017 --net mongo-cluster mongo --replSet replicaset_test f575196f3be169f2c1210b8e91bd2a8f5e06cbeea27f599920b7aafa0b40faa5
PS C:\Users\razva> docker run -d --name mongo-node2 -p 27801:27017 --net mongo-cluster mongo --replSet replicaset_test 4de7ff7cd5dd561ac3d7cf88166247e4196d5a547a8db2d8f77057c51fb398a
PS C:\Users\razva> docker run -d --name mongo-node3 -p 27802:27017 --net mongo-cluster mongo --replSet replicaset_test b6fb863ff03c2afd3e0159b0c3605c1454151d91156ab27472e6fbd8803254a6
PS C:\Users\razva>
```

3. Connect to any node, e.g. node #1 mongosh mongodb://localhost:27800/

```
PS C:\Users\razva> mongosh mongodb://localhost:27800/
Current Mongosh Log ID: 66492edeee9658ag769194b
Current Mongosh Log ID: 66492edeee9658ag769194b
Connecting to: mongodb://localhost:27800/deirectConnection=trueServerSelectionTimeoutMS=2000&appName=mongosh+2.0.0
Using Mongosh: 7.0.9
Using Mongosh: 2.0.0
mongosh 2.2.6 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

----
The server generated these startup warnings when booting
2024-05-18722-41:47.736-00:00: Using tho XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2024-05-18722-41:48, 2944+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'almays'. We suggest setting it to 'never' in this binary version

2024-05-18722:41:48, 2944+00:00: wm.max_map_count is too low

test>
```

4. Initiate the replica then check the replica status

At first, we create a JSON object that contains all members of the replica set

Then we initiate the replica set using the configuration we've just created > rs.initiate(config)

```
}
test> rs.initiate(config)
{ ok: 1 }
replicaset_test [direct: secondary] test>
```

At the end, check the replica set status and see that one node is PRIMARY and the rest are secondary

> rs.status()

```
eplicaset_test [direct: secondary] test> rs.status()
 set: 'replicaset_test'
 date: ISODate("2024-05-18T22:44:26.502Z"),
 myState: 1,
term: Long("1"),
 syncSourceHost:
 syncSourceId: -1
 héartbeatIntervalMillis: Long("2000"),
 majorityVoteCount: 2,
writeMajorityCount: 2
 votingMembersCount: 3
 writableVotingMembersCount: 3,
 optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1716072265, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2024-05-18T22:44:25.062Z"),
   readConcernMajorityOpTime: { ts: Timestamp({ t: 1716072265, i: 1 }), t: Long("1") }, appliedOpTime: { ts: Timestamp({ t: 1716072265, i: 1 }), t: Long("1") }, durableOpTime: { ts: Timestamp({ t: 1716072265, i: 1 }), t: Long("1") }, lastAppliedWallTime: ISODate("2024-05-18T22:44:25.062Z"),
    lastDurableWallTime: ISODate("2024-05-18T22:44:25.062Z")
 lastStableRecoveryTimestamp: Timestamp({ t: 1716072234, i: 1 }),
 electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout'
    lastElectionDate: ISODate("2024-05-18T22:44:04.961Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1716072234, i: 1 }), t: Long("-1")
lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1716072234, i: 1 }), t: Long("-1") },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    numCatchUpOps: Long("0"),
newTermStartDate: ISODate("2024-05-18T22:44:05.025Z"),
    wMajorityWriteAvailabilityDate: ISODate("2024-05-18T22:44:05.542Z")
 members: [
    {
      _id: 0,
name: 'mongo-node1:27017',
      health: 1,
      state: 1,
stateStr: 'PRIMARY',
      uptime: 159,
optime: { ts: Timestamp({ t: 1716072265, i: 1 }), t: Long("1") },
optimeDate: ISODate("2024-05-18T22:44:25.000Z"),
lastAppliedWallTime: ISODate("2024-05-18T22:44:25.062Z"),
      lastDurableWallTime: ISODate("2024-05-18T22:44:25.062Z"),
       syncSourceHost: '
      syncSourceId: -1, infoMessage: 'Could not find member to sync from',
      electionTime: Timestamp({ t: 1716072244, i: 1 }), electionDate: ISODate("2024-05-18T22:44:04.000Z"),
      configVersion: 1,
      configTerm: 1,
      self: true,
lastHeartbeatMessage: ''
      _id: 1,
name: 'mongo-node2:27017',
      health: 1,
      state: 2,
stateStr: 'SECONDARY',
```

5. Use the following connection string to connect to the MongoDB replica-set. Notice that we specify three hosts instead on one.

## mongosh

mongodb://localhost:27800,localhost:27801,localhost:27802/repldb?directConnection=true

```
6. Let's do an insert into the replica set:
```

## DO IT YOURSELF:

A. Insert some actor names using the "majority" write-concern and a timeout of 2 seconds.

## NEXT:

We will explore time series collections. Familiarize yourself with the topic at https://docs.mongodb.com/manual/core/timeseries-collections/