NBD MongoDB Distribution lab. 1

Create 3 directories in mongo/bin directory:

- data\db1\
- data\db2\
- data\db3\

Type given commands accurately.

Mind that after restarting Mongo server there can be problems with sending commands from mongo shell. If it happens, enter the command again to make it work.

Start 3 instances of mongo servers (mongod), each in a separate command line terminal:

```
mongod --port 27017 --dbpath .\data\db1 --replSet replicaSet
mongod --port 27018 --dbpath .\data\db2 --replSet replicaSet
mongod --port 27019 --dbpath .\data\db3 --replSet replicaSet
```

Those servers (and connected clients) will be identified by db1, db2, db3 respectively.

In a new command line terminal connect to db1 server by running command:

```
mongosh --host "127.0.0.1:27017"
```

All further commands with *"db1->"* prefix should be executed in this command line client.

Configuring replica set:

```
db1->rs.initiate()          //initializing replica set with default configuration
db1->cfg = rs.conf()        //retrieving the configuration
db1->cfg.members[0].host = '127.0.0.1:27017'  //changing host address of first node
db1->rs.reconfig(cfg)               //applying changed configuration
db1->rs.add('127.0.0.1:27018')              //adding a normal node
db1->db.adminCommand({getDefaultRWConcern:1})    //getting default read concern
// it's "majority", 1 will be needed to add arbiter
db1->db.adminCommand({setDefaultRWConcern:1, defaultWriteConcern: {w:1}})         //setting required value
db1->rs.addArb('127.0.0.1:27019')                   //adding arbiter
//if configuration with all members was provided to rs.initiate(), changing default write concern would not be needed
```

Tests:

```
db1->rs.status()
db1->db.isMaster()                  //this command won't show hidden nodes
db1->db.mycol.insertOne({field:'value'})
db1->db.mycol.find()
```

Connecting to other two mongo servers:

In a new command line terminal connect to db2 server by running command:
mongosh --host "127.0.0.1:27018"

In a new command line terminal connect to db3 server by running command:
mongosh --host "127.0.0.1:27019"

db2->db.mycol.find()
db3->db.mycol.find()
Commands return error, because for now we can’t read from secondary.

Loosing a primary node:
Turn off db1 server.
db1->db.mycol.find()
db2-> rs.status()
db2->db.mycol.find()
Turn db1 server back on.
db1->rs.status() //db1 and db2 have the same priority, db2 remain primary
db1->db.mycol.find()

Reading from secondary:
db2-> db.getMongo().setReadPref('primaryPreferred')
db2->db.mycol.find()
db1-> db.getMongo().setReadPref('primaryPreferred')
db1->db.mycol.find()
db3-> db.getMongo().setReadPref('primaryPreferred')
db3->db.mycol.find() //only now db3 server “remembers” it is an arbiter

Priorities:
db2->cfg = rs.conf()
db2->cfg.members[0].priority = 2 //setting db1’s priority to 2
db2->rs.reconfig(cfg)
db2->rs.status() //wait a few seconds before running this command

Loosing primary with higher priority:
Turn of db1 server and wait a few seconds.
db2->rs.status()
Now turn it back on and wait a moment.
db2->rs.starus()

Priority 0:
db1->cfg = rs.conf()
db1->cfg.members[1].priority = 0 //Setting db2 priority to 0
db1->rs.reconfig(cfg) //reconfiguring replica set
db1->rs.conf()

Loosing primary with secondary’s priority being 0:
Turn of db1 server.
db2->rs.status()

db2->db.mycol.find()
db2->db.mycol.insertOne({field2:"value2"})
Turn on db1 server again.
db1->rs.status()
db1->db.mycol.find()
db1->db.mycol.insertOne({field2:"value2"})

Write concern:
Turn off db2 server.
db1->rs.status()
db1->db.mycol.insertOne({a:"a"})
db1->db.adminCommand({setDefaultRWConcern : 1, defaultWriteConcern: { w: *"majority"* }})
//changing default write concern to majority
db1->db.mycol.insertOne({b:[1,3]}) //here the client should freeze
//turn off the db1 client and turn it on again.
db1->db.adminCommand({setDefaultRWConcern : 1, defaultWriteConcern: { w: *"majority"*, wtimeout: 5000 }, writeConcern: { w: 1 }}) //5000 miliseconds timeout; default write concern is "majority", so for the command to succeed we need to set write concern for this command to 1
db1->db.mycol.insertOne({b:[3,1]})
Turn on db2 server.
db2->db.mycol.find()

Delayed node:
db1->cfg = rs.conf() *//preparing configuration for db3 server (by altering current member configuration)*
db1->cfg.members[2].arbiterOnly = false
db1->cfg.members[2].priority = 0 *//delayed member must have priority 0 (cannot become primary)*
db1->cfg.members[2].secondaryDelaySecs = 15 //seconds
db1->cfg.members[2].votes = 0 //will not vote
db1->cfg.members[2].hidden = true // will be hidden
db1->rs.remove("127.0.0.1:27019") //cannot reconfigure arbiter
db1->rs.add(cfg.members[2])
db1->rs.status()
db3->db.mycol.find()
db3-> db.getMongo().setReadPref('primaryPreferred')
db3->db.mycol.find()
db1->db.isMaster() *//this command won't show hidden nodes*
db3->db.isMaster() *//even on the hidden node*
db1->db.mycol.findOneAndReplace({field:"value"},{ddd:3})
db1->db.mycol.find()
db3->db.mycol.find()

Individual task

Configure a replica set with the following members:

- 2 nodes with priority 3
- 1 node delayed by an hour (voting)
- 2 non-voting nodes

After completing the task submit the following info:

- configuration of the replica set
- status of the replica set.