Replication Notes

Starting a Master/Slave Pair

Create directories for each mongod instance:

```
mkdir /data/master /data/slave
```

Start the master and the slave:

```
mongod --dbpath /data/master --port 27017 --master

# You may have to replace "localhost" with your machine's hostname.
mongod --dbpath /data/slave --port 27018 --slave --source localhost:27017
```

Starting a Replica Set

Create directories for each mongod instance:

```
mkdir /data/rs0 /data/rs1 /data/rs2
```

Start three instances of mongod:

```
mongod --dbpath /data/rs0 --port 27017 --replSet myset

mongod --dbpath /data/rs1 --port 27018 --replSet myset

mongod --dbpath /data/rs2 --port 27019 --replSet myset
```

Connect to the any mongod instance:

```
mongo --port 27018
```

Initialize the replica set:

```
> rs.initiate()
```

After a few seconds the mongod process you are connected to will become the replica set primary. The prompt will change to "PRIMARY>". Run rs.status() to see the current configuration:

```
PRIMARY> rs.status()
{
      "set" : "myset",
      "date" : ISODate("2012-05-11T17:56:37Z"),
      "myState" : 1,
      "members" : [
              {
                      "_id" : 0,
                      "name" : <your hostname>:27018",
                      "health" : 1,
                      "state" : 1,
                      "stateStr" : "PRIMARY",
```

```
                        "optime" : {
                                "t" : 1336758831000,
                                "i" : 1
                        },
                        "optimeDate" : ISODate("2012-05-11T17:53:51Z"),
                        "self" : true
                }
        ],
        "ok" : 1
}
```

Add the other members:

```
PRIMARY> rs.add('<your hostname>:27017')
PRIMARY> rs.add('<your hostname>:27019')
```

Config Objects

There are many optional settings that can be configured using the config object:

```
{
  _id : <setname>,
  members: [
    {
      _id : <ordinal>,
      host : <hostname[:port]>
      [, arbiterOnly : true]
      [, buildIndexes : <bool>]
      [, hidden : true]
      [, priority: <priority>]
      [, tags: {loc1 : desc1, loc2 : desc2, ..., locN : descN}]
      [, slaveDelay : <n>]
      [, votes : <n>]
    }
    , ...
  ],
  [settings: {
    [getLastErrorDefaults: <lasterrdefaults>]
    [, getLastErrorModes : <modes>]
  ]
}
```

A quick example:

```
> conf = { _id: 'myset',
...        members: [
...          { _id: 0, host: '<your hostname>:27017'},
...          { _id: 1, host: '<your hostname>:27018'},
...          { _id: 2, host: '<your hostname>:27019'}
...        ]
...      }

> rs.initiate(conf)
```

To reconfigure the set:

```
PRIMARY> conf = rs.conf()
PRIMARY> conf.members[2].priority = 100
PRIMARY> rs.reconfig(conf)
```

To remove an option set it to its default setting:

```
PRIMARY> conf = rs.conf()
PRIMARY> conf.members[2].priority = 1
PRIMARY> rs.reconfig(conf)
```

Other Important Commands

```
rs.help()
rs.status()
rs.slaveOk()
db.printReplicationInfo()
db.printSlaveReplicationInfo()
```

Exercises

1. Set up a replica set using the steps above.
2. Run the command to step down the primary: db.runCommand({ replSetStepDown: 1}); Ensure that a secondary node is elected as the new primary.
3. Practice automated failover. In this case, you'll want to terminate the primary node manually.
4. Add a node to an existing live replica set. This involves setting up a new node and either running rs.add() from the shell or, on a lower level, running the replicaSetReconfig command.