USING MONGODB REPLICATION

Version: 1.0

Author: Khoa Truong

TABLE OF CONTENTS

1. Introduction 2
2. Convert a standalone to a replica set 4
3. See also 6

1. INTRODUCTION

A *replica set* in MongoDB is a group of *mongod* processes that maintain the same data set. Replica sets provide redundancy and high availability, and are the basis for all production deployments.

A replica set contains several data bearing nodes and optionally one arbiter node. Of the data bearing nodes, one and only one member is deemed the primary node, while the other nodes are deemed secondary nodes.

The primary node receives all write operations. If the primary is unavailable, an eligible secondary will hold an election to elect itself the new primary.

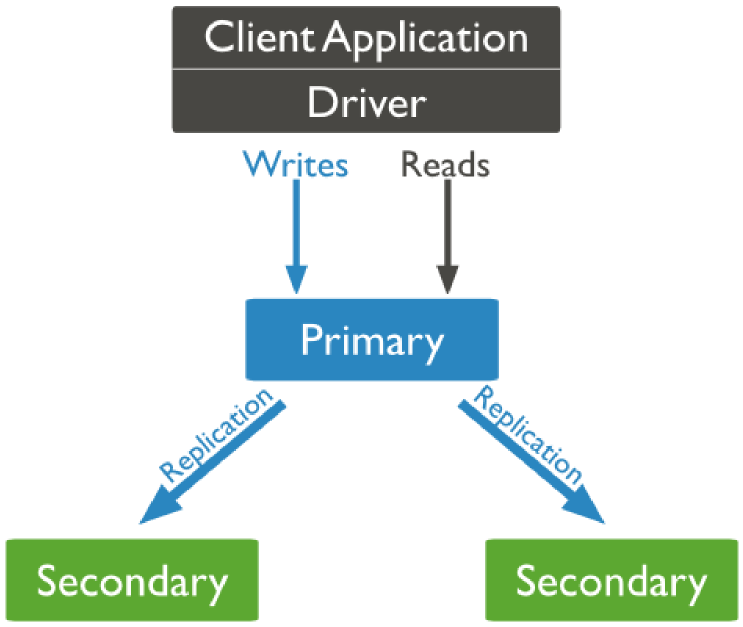

Figure 1 – Members of a replica set

You may add an extra *mongod* instance to a replica set as an ***arbiter***. The purpose of an arbiter is to maintain a quorum in a replica set by responding to heartbeat and election requests by other replica set members.

Arbiters do not maintain a data set.

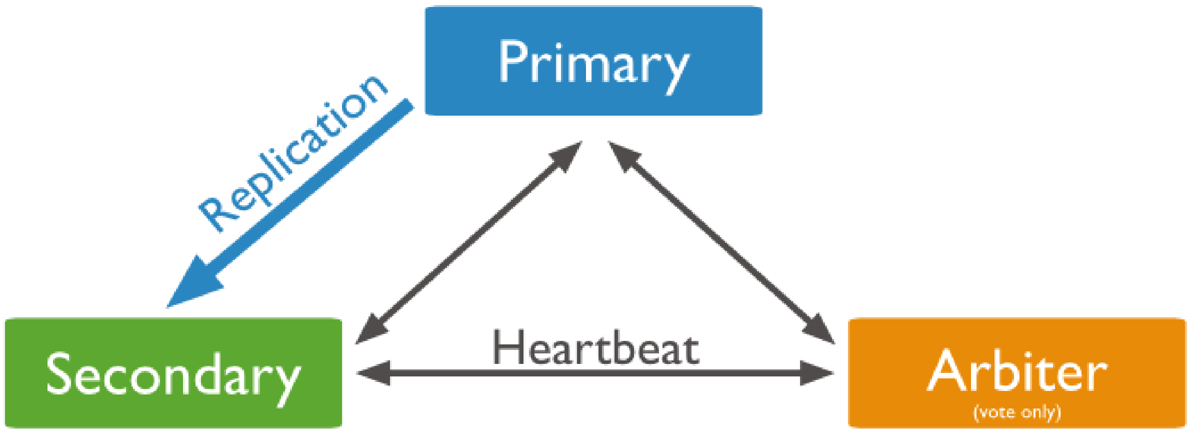

Figure 2 - Replica set with primary, secondary and arbiter

2. CONVERT A STANDALONE TO A REPLICA SET

Example:

Given a replica set with three members running on three separate hosts, we'll also follow the Figure 2 - Replica set with primary, secondary and arbiter to setup the replica set:

- m1.example.net (which includes the existed standalone instance, it will be primary member)
- m2.example.net (which includes a *mongod* instance as secondary member)
- m3.example.net (which includes a *mongod* instance as secondary member)

Procedure:

2.1. On the host *m1.example.net*, shutdown the standalone instance.

2.2. Restart the instance with a specified name of replica set

Uses the ***--replSet*** option to specify the name of the new replica set.

```
mongod --port 27017 --replSet rs1
```

If start the instance using configuration file, please refer to Configuration file options

2.3. Initiate the new replica set

On the host *m1.example.net*, start the MongoDB shell, type:

```
rs.initiate()
```

Now, the instance on the host *m1.example.net* become the primary

*To view the replica set configuration, use **rs.conf()***

*To check the status of the replica set, use **rs.status()***

2.4. Expand the replica set

Setup MongoDB on the two hosts *m2.example.net* and *m3.example.net*, then **start** the mongod instance on each host:

```
mongod --port 27017 --replSet rs1
```

To add members to the replica set, open MongoDB shell which connect with the primary member on the host *m1.example.net*.

Add the secondary member to the replica set:

```
rs.add("m2.example.net:27017")
```

Add the arbiter to the replica set:

```
rs.addArb("m3.example.net:27017")
```

The name of replica set must be same when start mongod instances, in our case it's "rs1".

Should deploy an odd number of members.

2.5. Connect to the replica set

To connect to the replica set, use the connection string:

```
mongodb://m1.example.net:27017,m2.example.net:2
7017/databaseName?replicaSet=rs1
```

Standard Connection String Format

3. SEE ALSO

- Deploy an odd number of members:
  - Ensure that the replica set has an odd number of voting members. If you have an even number of voting members, deploy an arbiter so that the set has an odd number of voting members.
  - In general, avoid deploying more than one arbiter per replica set.
- Maximum Number of Voting Members:
  - A replica set can have up to 50 members, but only 7 voting members.
  - If the replica set already has 7 voting members, additional members must be non-voting members.
- **Replica set deployment architectures**
- Replica set elections
- Wikipedia - Quorum (distributed computing)