g·able

MongoDB 3.0

Replication

Topics

- Replication
- Labs

Replication

- Replication is the process of synchronizing data across multiple servers.

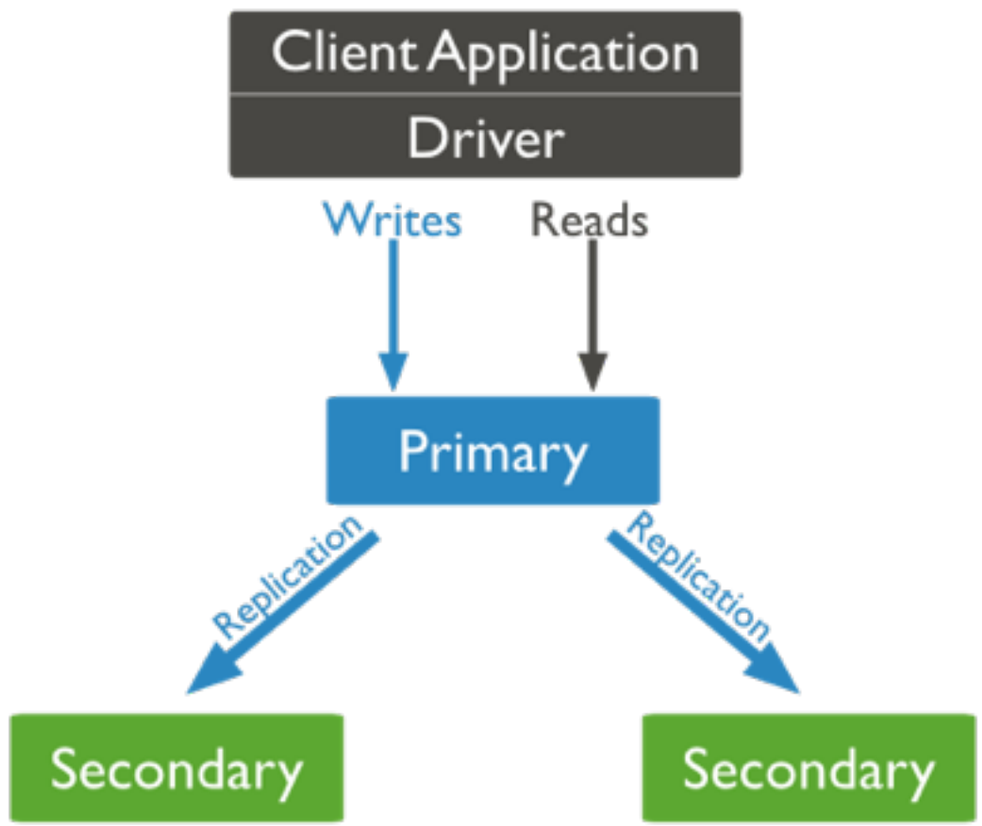
- Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication protects a database from the loss of a single server.

- Replication also allows you to recover from hardware failure and service interruptions. With additional copies of the data, you can dedicate one to disaster recovery, reporting, or backup.

- In some cases, you can use replication to increase read capacity. Clients have the ability to send read and write operations to different servers. You can also maintain copies in different data centers to increase the locality and availability of data for distributed applications.

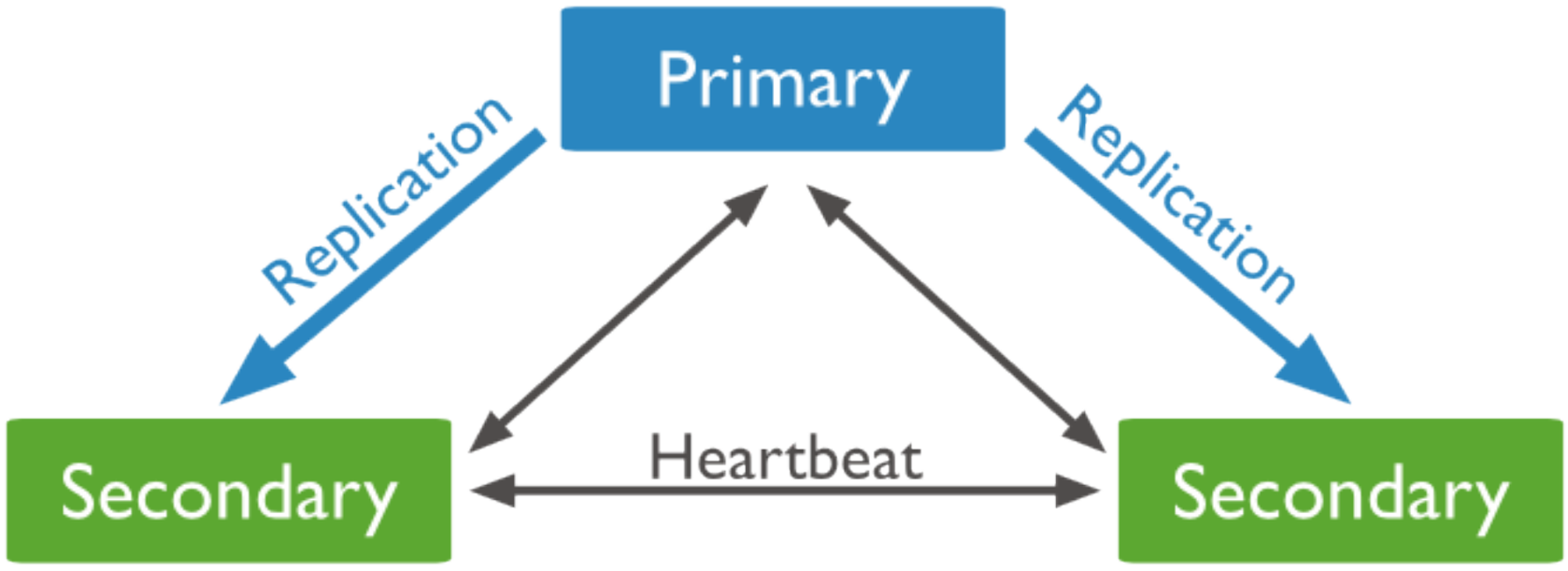
Replication in MongoDB

- A replica set is a group of mongod instances that host the same data set. One mongod, the primary, receives all write operations. All other instances, secondaries, apply operations from the primary so that they have the same data set.



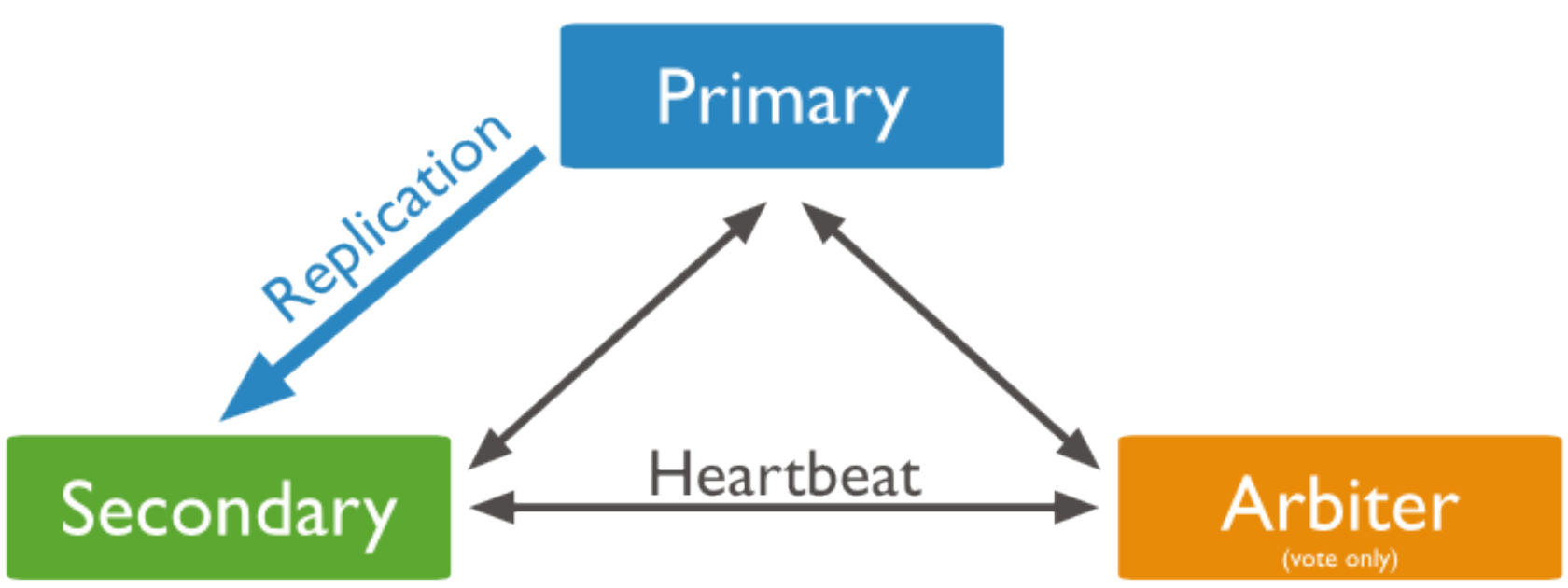
Replication in MongoDB

- The secondaries replicate the primary's oplog and apply the operations to their data sets. Secondaries' data sets reflect the primary's data set. If the primary is unavailable, the replica set will elect a secondary to be primary. By default, clients read from the primary, however, clients can specify a read preferences to send read operations to secondaries. Reads from secondaries may return data that does not reflect the state of the primary. See secondaries for more information.



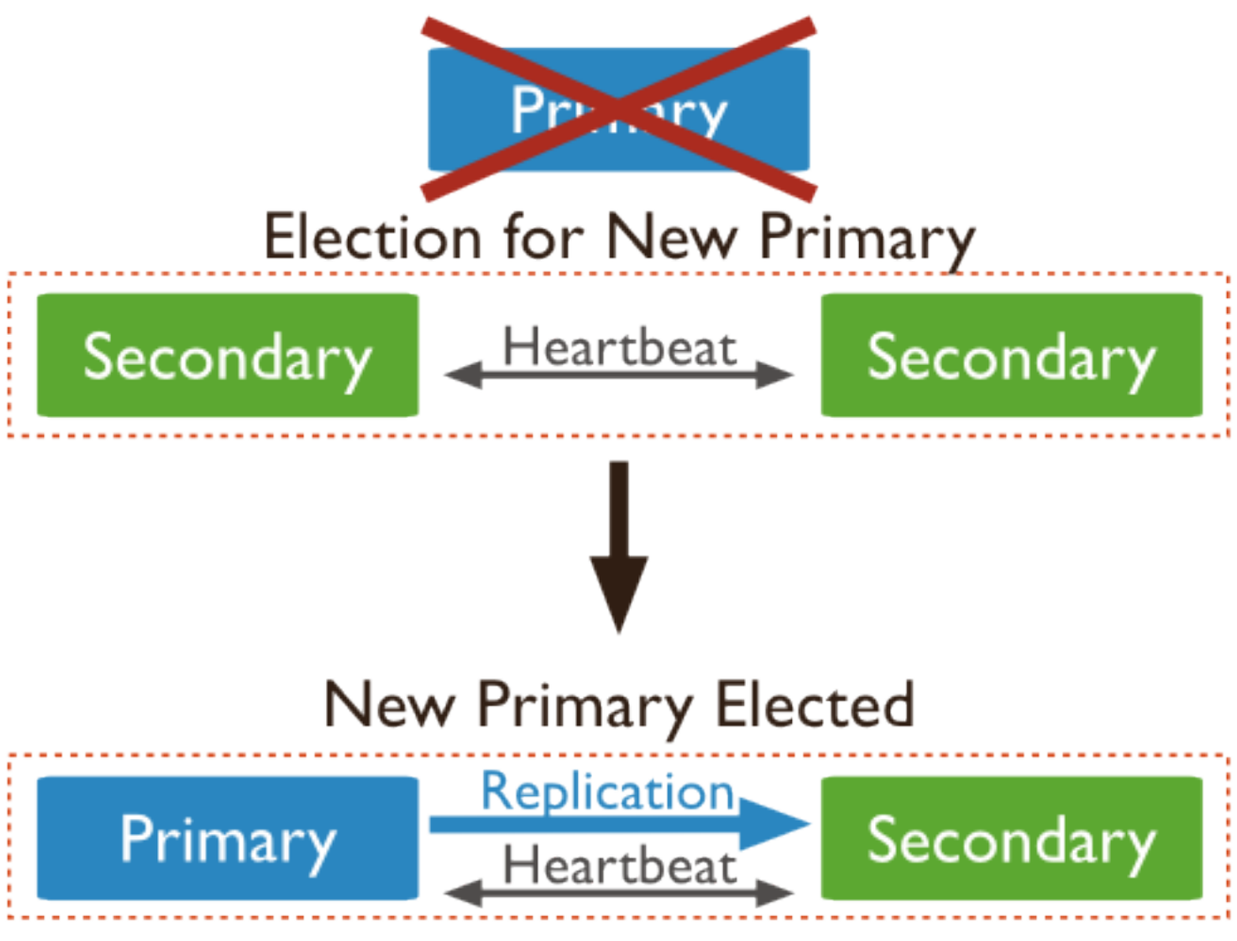
Replication in MongoDB

- You may add an extra mongod instance to a replica set as an arbiter. Arbiters do not maintain a data set. Arbiters only exist to vote in elections. If your replica set has an even number of members, add an arbiter to obtain a majority of votes in an election for primary. Arbiters do not require dedicated hardware. See arbiter for more information.



Automatic Failover

- When a primary does not communicate with the other members of the set for more than 10 seconds, the replica set will attempt to select another member to become the new primary. The first secondary that receives a majority of the votes becomes primary.



Lab1: Prepare Replication

1. Create 3 folders on the desktop, named “R1Primary”, “R1Secondary”, and “R1Arbiter”.

2. Open Terminal. Issuing the following command to stop default MongoDB instance.

   ```
   sudo service mongod stop
   ```

3. Open new Terminal. Issuing the following command to start primary instance for MongoDB Replication Set.

   ```
   mongod --port 27018 --dbpath /home/mongo-root/Desktop/
   R1Primary/ --replSet rs0 --smallfiles --oplogSize 128
   ```

   Note: The *--smallfiles* and *--oplogSize* settings reduce the disk space that each mongod instance uses. This is ideal for testing and development deployments as it prevents overloading your machine.

Lab1: Prepare Replication

4. Leave the Terminal open and open the second Terminal. Type the following command to start the secondary instance.

```
mongod --port 27019 --dbpath /home/mongo-root/
Desktop/R1Secondary/ --replSet rs0 --
smallfiles --oplogSize 128
```

5. Open the third Terminal. Type the following command to start the arbiter instance.

```
mongod --port 27020 --dbpath /home/mongo-root/
Desktop/R1Arbiter/ --replSet rs0 --smallfiles
--oplogSize 128
```

Lab1: Initial Replication Set

6. Open the forth Terminal. Connect to Primary MongoDB instant.

```
mongo --port 27018
```

7. On Mongo shell, type the following command to initial replication set.

```
rsconf = {
    _id: "rs0",
    members: [{_id: 0,host: "127.0.0.1:27018"}]
}

rs.initiate(rsconf)
```

8. Display the current replica configuration by issuing the following command:

```
rs.conf()
```

Lab1: Add Secondary & Arbiter

9. In the mongo shell connected to the primary, add the second mongod instance to the replica set using the rs.add() method.

   ```
   rs.add("127.0.0.1:27019")
   ```

10. Add arbiter instances by type the following command:

    ```
    rs.addArb("127.0.0.1:27020")
    ```

11. Display the current replica configuration again by issuing the following command. You will see 3 members in JSON response document for primary, secondary, and arbiter.

    ```
    rs.conf()
    ```

12. Issuing the following command to check is this instance is primary or not:

    ```
    db.isMaster()
    ```

Lab2: Import data

1. Open New Terminal. Goto Desktop folder.

   ```
   cd /home/mongo-root/Desktop/
   ```

2. Issuing the following command to import data to MongoDB Replication Set.

   ```
   mongoimport --port 27018 --db twitter_data --collection tweets --file
   tweets.json
   ```

3. Verify imported data by use mongo shell to connect to primary. Remember the number of documents after issued the following commands:

   ```
   mongo --port 27018

   show dbs
   use twitter_data
   show collections
   db.tweets.count()
   ```

4. Exit from primary instance. Then connect mongo shell to the secondary instance.

   ```
   mongo --port 27019
   ```

Lab2: Verify Replicated Data

5. Try to issue the following command. You will see “listed database fail” exception:

   ```
   show dbs
   ```

   Note: By default, slaves are for backup only, but you can also use them for queries (reads) if you set the “slave ok” flag.

6. Issuing the following command to enable read from secondary instance.

   ```
   db.getMongo().setSlaveOk()
   ```

7. Now, issuing the following commands to see the documents within the secondary database. The number showing from the latest command should be the same with the primary instances.

   ```
   use twitter_data
   show collections
   db.tweets.count()
   ```

Lab3: Failure Test

1. Switch to the Terminal that running MongoDB primary replication set. Terminate the process by press Control+C on keyboard.

2. Open the new Terminal. Connect to the secondary MongoDB instance with mongo shell.

   ```
   mongo --port 27019
   ```

   Now you can see the prompt showing this instance is become primary.

3. Go back to the Terminal in step 1. Run mongod process to get the primary back again.

   ```
   mongod --port 27018 --dbpath /home/mongo-root/Desktop/R1Primary/ --
   replSet rs0 --smallfiles --oplogSize 128
   ```

4. From Terminal on step 2, exit mongo shell and connect to the old primary instances.

   ```
   mongo --port 27018
   ```

   Now you can see this instance become secondary.

References

- Replication Introduction
  http://docs.mongodb.org/manual/core/replication-introduction/

- Tutorial & Deploy
  http://docs.mongodb.org/manual/tutorial/deploy-replica-set/