MongoDB Replica Set Tutorial: Step by Step Replication Example

Details

Last Updated: 03 January 2020

What is MongoDB Replication?

Replication is referred to the process of ensuring that the same data is available on more than one Mongo DB Server. This is sometimes required for the purpose of increasing data availability.

Because if your main MongoDB Server goes down for any reason, there will be no access to the data. But if you had the data replicated to another server at regular intervals, you will be able to access the data from another server even if the primary server fails.

Another purpose of replication is the possibility of load balancing. If there are many users connecting to the system, instead of having everyone connect to one system, users can be connected to multiple servers so that there is an equal distribution of the load.

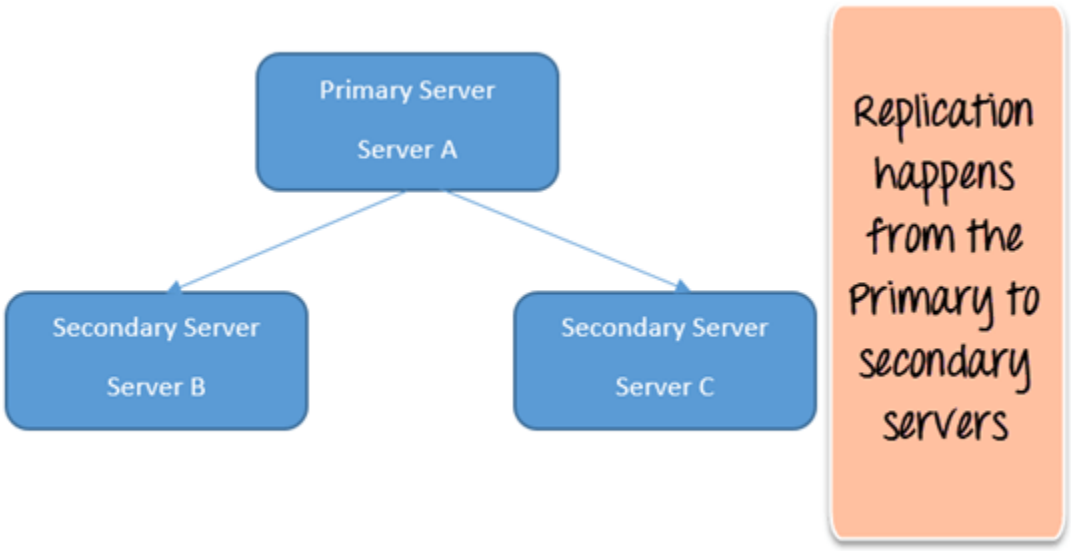
In MongoDB, multiple MongDB Servers are grouped in sets called Replica sets. The Replica set will have a primary server which will accept all the write operation from clients. All other instances added to the set after this will be called the secondary instances which can be used primarily for all read operations.

In this tutorial, you will learn –

Replica Set: Adding the First Member using rs.initiate()

As mentioned in the previous section, to enable replication, we first need to create a replica set of MongoDB instances.

Let's assume that for our example, we have 3 servers called ServerA, ServerB, and ServerC. In this configuration, ServerA will be our Primary server and ServerB and ServerC will be our secondary servers. Below screenshot will give a better idea on it.



Below are the steps which need to be followed for creating the replica set along with the addition of the first member to the set.

Step 1) Ensure that all mongod.exe instances which will be added to the replica set are installed on different servers. This is to ensure that even if one server goes down, the others will be available and hence other instances of MongoDB will be available.

Step 2) Ensure that all mongo.exe instances can connect to each other. From ServerA, issue the below 2 commands

```
mongo –host ServerB –port 27017
mongo –host ServerC –port 27017
```

Similarly, do the same thing from the remaining servers.

Step 3) Start the first mongod.exe instance with the replSet option. This option provides a grouping for all servers which will be part of this replica set.

```
mongo –replSet "Replica1"
```

Where "Replica1" is the name of your replica set. You can choose any meaningful name for your replica set name.

Step 4) Now that the first server is added to the replica set, the next step is to initiate the replica set by issuing the following command rs.initiate ()

Step 5) Verify the replica set by issuing the command rs.conf() to ensure the replica set up properly

Replica Set: Adding a Secondary using rs.add()

The secondary servers can be added to the replica set by just using the rs.add command. This command takes in the name of the secondary servers and adds the servers to the replication set.

Step 1) Suppose if you have ServerA, ServerB, and ServerC, which are required to be part of your replica set and ServerA, is defined as the primary server in the replica set.

To add ServerB and ServerC to the replica set issue the commands

```
rs.add("ServerB")
rs.add("ServerC")
```

Replica Set: Reconfiguring or Removing using rs.remove()

To remove a server from the configuration set, we need to use the "rs.remove" command

Step 1) First perform a shutdown of the instance which you want to remove. One can do this by issuing the db.shutdownserver command from the mongo shell.

Step 2) Connect to the primary server

Step 3) Use the rs.remove command to remove the required server from the replica set. So suppose if you have a replica set with ServerA, ServerB, and ServerC, and you want to remove ServerC from the replica set, issue the command

```
rs.remove("ServerC")
```

Troubleshooting Replica Sets

The following steps are same ways one can troubleshoot when issues are encountered with the usage of replica sets.

1. Ensure that all mongo.exe instances can connect to each other. Suppose if you have 3 servers called ServerA, ServerB, and ServerC. From Server A, issue the below 2 commands

```
mongo –host ServerB –port 27017
mongo –host ServerC –port 27017
```

2. Run the rs.status command. This command gives the status of the replica set. By default, each member will send messages to each other called "heartbeat" messages which just indicates that the server is alive and working. The "status" command get the status of these messages and shows if there are any issues with any members in the replica set.
3. Check the size of the Oplog – The Oplog is a collection in MongoDB that stores the history of writes which were done to the MongoDB database. MongoDB then uses this Oplog to replicate the writes to the other members in the replica set. To check the Oplog, connect to the required member instance and run the rs.printReplicationInfo command. This command will show the size of the log and how long it can hold transactions in its log file before it becomes full.

Summary:

- Replication is referred to the process of ensuring that the same data is available on more than one Mongo DB Server. Many members (MongoDB instances) can be added to the Replica set depending on the requirements.