Install a mongodb Replica Set: 1 Primary Node and 2 Secondary Nodes

Fernando G. Tinetti*, Agustín Terruzzi

Technical Report TR-BDD-01-2017
III-LIDI, Fac. de Informática, UNLP
*Also with CIC, Prov. de Buenos Aires
La Plata, Argentina
contact e-mail: fernando@info.unlp.edu.ar
March 2017

Abstract. In this report, we document the installation a mongodb replica set for an initial environment of 3 nodes: 1 primary node and 2 secondary nodes from scratch. Even when there are many web sites and tutorials on mongodb replica sets, we did not find anyone simple enough for a simple environment, as well as with some methodological way of going from a non-replicated environment to a replicated one and vice-versa. Having to go back and forth from replicated to non-replicated environment enables several experiments on which we are interested in. We are particularly interested in an environment focused on the replica set (functionality and runtime) performance for analysis not affected by other details such as web applications, schema control of mongodb NoSQL databases, and web development frameworks, which are usually included in most (it not all) mongodb tutorials.

1.- Introduction

Our main goal is to have a step-by-step guide for having a three-node replicated mongo NoSQL database, as shown in Fig. 1 [1]. In terms of mongo replication, we configure a three-node replica set, where one of them is the so called Primary node (i.e. the authoritative and up-to-date one) and the two others are (peer) Secondary nodes, which replicate the Primary node write operations [2].

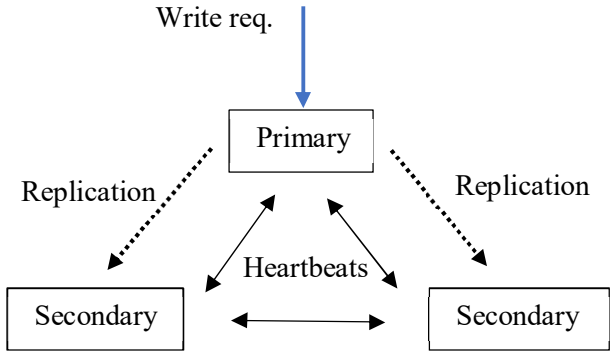

Figure 1: Three-node Replica Set, One Primary Node, ad Two Secondary Nodes.

We will use virtual machines (vm) in order to build up the whole replica set system, so that the environment is closer to a real one, in which the actual nodes usually are vm too. However, we will use a single host

computer in which the three virtual guests will run. Initially, the host computer will provide only connectivity (networking), but it could be later used for running a client in a fourth/different computer, other than the three node replica set. A 64-bit operating system will be installed in every vm, since it is required by the current mongodb version. Actually, we use a standard distribution: a 64-bit Ubuntu (the current one at the time of this writing). VirtualBox will be used for handling the vm guests, and the vm network guest will be configured as the so called "Bridged Adapter" one, so that the host computer will be in the same local network as the guest vm, and every vm has access to the internet. Even when the description below is made as if every computer is individually handled (for a clear explanation), many of the steps can be carried out in a single installation (i.e. a "reference install") and that vm can be later cloned. Cloning a vm is easy in every virtual computer manager, as in VirtualBox. Maybe the most time-consuming step would be the operating system installation, so we suggest the OS is installed in a single vm and that vm cloned for the corresponding individual configuration (network interface, mongodb replica set node, etc.). In the end, a cluster-like environment is built and deployed for a mongo replica set experimentation.

2.- Network Settings

We will use a Windows computer as the host running the VirtualBox vm, so we take the host network configuration as the reference for the vm. Fig. 2 shows the actual host computer network configuration, where the IP and mask are used for defining those of the vm.

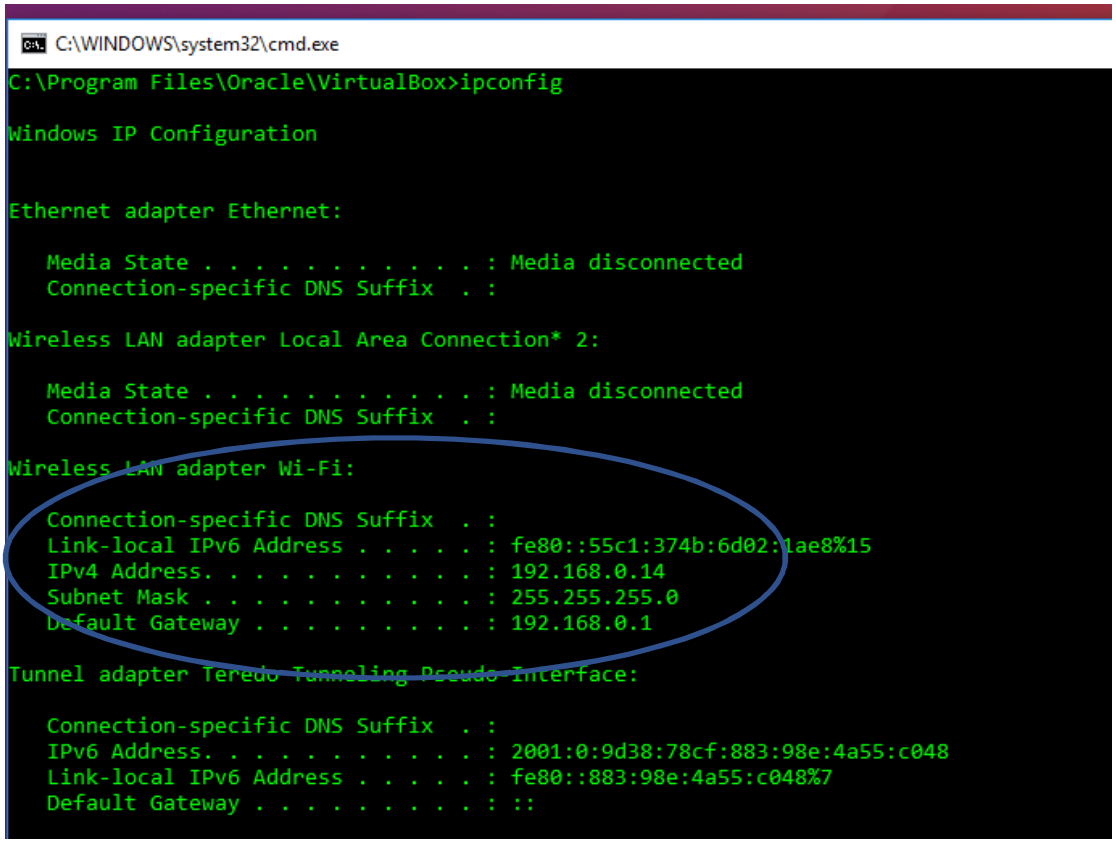

Figure 2: Actual Host Computer Network Configuration.

Since all the vm will have a Bridged Adapter, we will have into account the host IP in order to define the vm fixed IPs. In the context of a standard DHCP server for the computers, we should be careful to avoid IP conflicts. There are two options for avoiding IP conflicts: 1) Use a DHCP specific configuration for assigning fixed IPs to the vm, and 2) Assign each vm IP locally with conflicts low likelihood fixed IPs (we usually refer to such IPs as "far" from the host IP). We know option 2) is not completely safe, but it is certainly simpler than the option 1) so we will use it. Furthermore, the vm are relatively transient in that they are not database production computers, but computers used for specific experiments, so they are not expected to be always running. Fig. 3 to Fig. 5 show the sequence of choices and values for setting a fixed IP vm. Take into account that the host computer has been assigned the IP 192.168.0.14 and the vm is set to have the 192.168.0.160 IP number (with a low likelihood of being assigned by the DHCP server to another host in the same LAN, because it starts from "low" to "high" IP numbers).

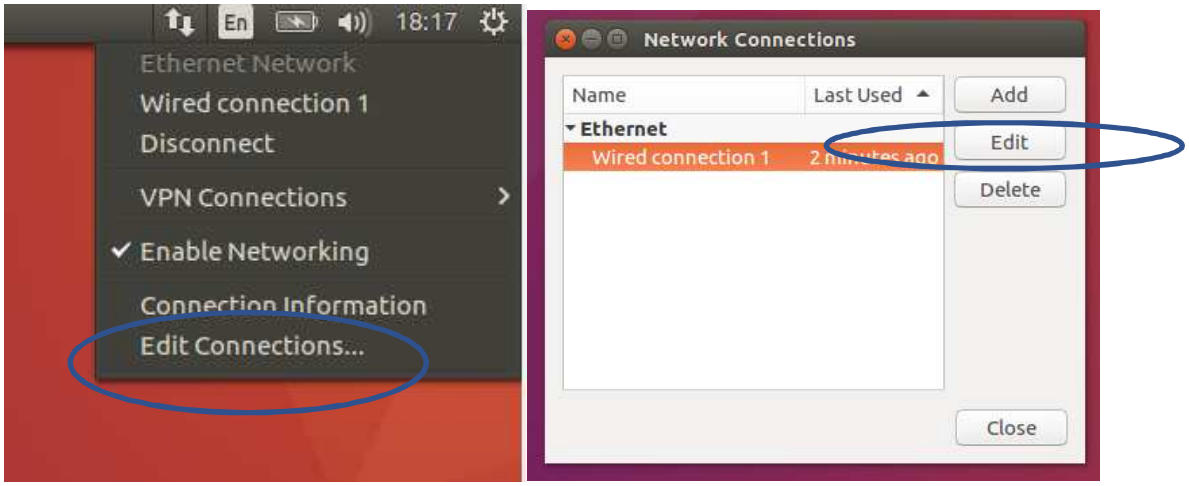

Figure 3: Edit Network Connection.

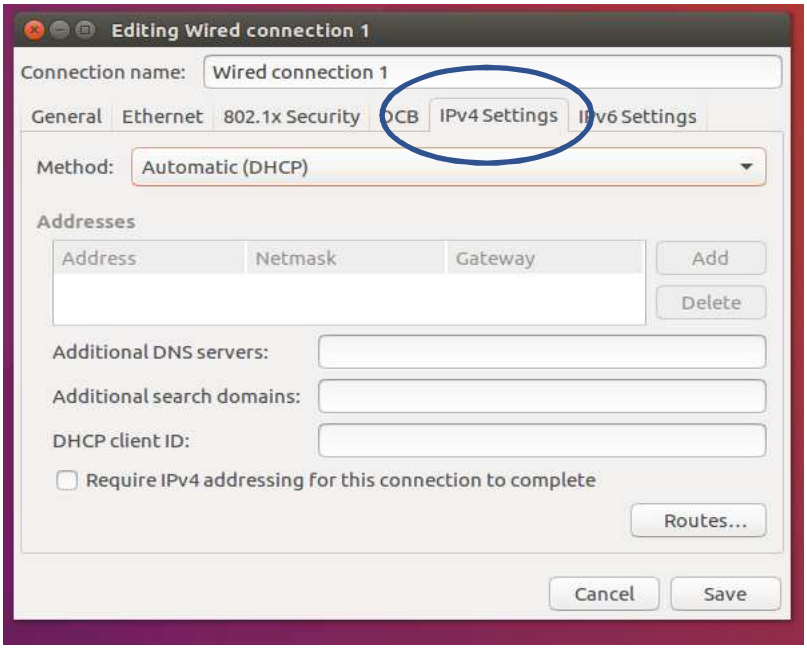

Figure 4: Edit/Set IPv4 Network Configuration.

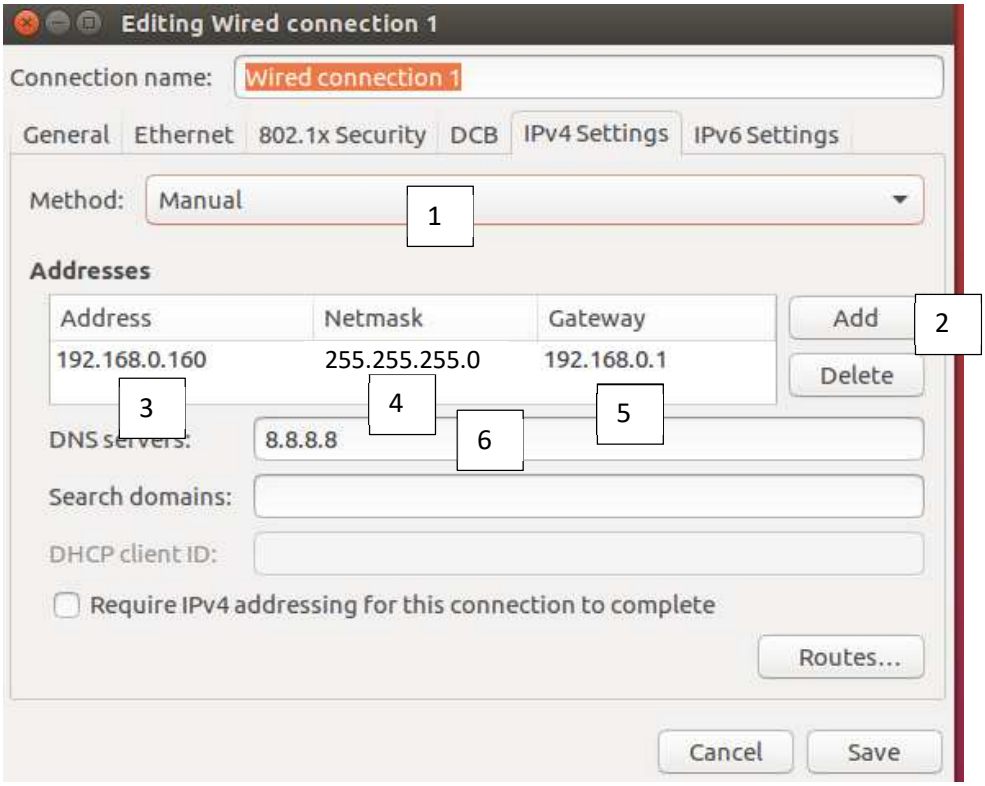

Figure 5: Specific IPv4 (IP) Values for the First vm.

All the vm (guests) IP and network settings are defined in terms of the host one, with a possible exception in the DNS server, which has been set to a well-known one, as shown in Fig. 5 above. Data set in the "Addresses" section in Fig. 5 need to end with the "Enter" key individually (sometimes the "Tab" key fails to set the correct value). Values for "Netmask" and "Gateway" are the same as the host computer, as shown in Fig. 2 above.

The second vm would have the same values for the Ethernet device, except for the IP value ("Address"), as shown in Fig. 6, which is exactly the same as Fig. 5 but the Address value: 192.168.0.161. Setting the third computer with its own IP, say 192.168.0.162, we will have:

- Three vm with static IP for their network connection
  - 192.168.0.160, 192.168.0.161, and 192.168.0.162 respectively
- Each vm has internet connection through the host computer
- Each vm is in the same LAN as its host (which has IP 192.168.0.14 as shown in Fig. 2)

The network settings have to be completed at the level of hostnames and local DNS resolution in order to avoid setting a local DNS server for name resolution. The hostnames have to be set in case a single installation is cloned, because the hostname will remain that of the original/cloned host otherwise. Local name resolution inconsistencies has to be fixed before starting the mongo replica set operation in order to avoid problems in mongo server reachability/visibility among them. Basically, check the following steps in each vm:

a) Set/verify the vm hostnames in /etc/hostname
b) Set the correct IPs in /etc/hosts

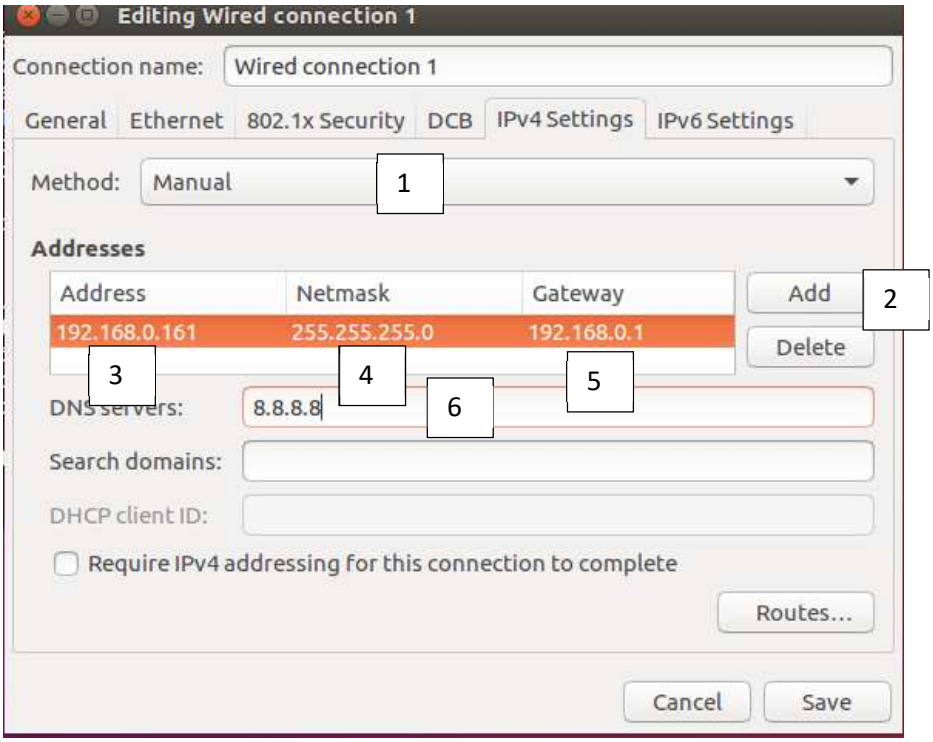

Figure 6: Specific IPv4 (IP) Values for the Second vm.

As a result of the previous steps, /etc/hostname and /etc/hosts in the first computer (that with IP 192.168.0.160) would be:

/etc/hostname
==========
distr2017
==========

/etc/hosts
==========
127.0.0.1 localhost

192.168.0.160 distr2017

192.168.0.161 distr2017r1
192.168.0.162 distr2017r2

Changed/added

The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
==========

The files in the second computer the second computer (that with IP 192.168.0.161) would be:

/etc/hostname
==========
distr2017r1
==========

/etc/hosts
==========
127.0.0.1 localhost

192.168.0.161 distr2017r1

192.168.0.160 distr2017
192.168.0.162 distr2017r2

Changed/added

The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
==========

And the files in the third computer (that with IP 192.168.0.162) would be:

/etc/hostname
==========
distr2017r2
==========

/etc/hosts
==========
127.0.0.1 localhost

192.168.0.162 distr2017r2

192.168.0.160 distr2017
192.168.0.161 distr2017r1

Changed/added

The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
==========

3.- Install mongodb and Configure the Replica Set

Installation of the mongodb NoSQL DBM is straightforward:
a) Install package:
    $ sudo apt-get install mongodb
b) Test the initial install:
    $ mongo
    (exit)

The replica set configuration is straightforward too, but involves several steps, as follows. The first step to configure the replica set is to stop the mongodb server (mongod process), because it should start configured for replication, which is not made by default (at the install step):
1) $ sudo service mongodb stop

The replica set configuration must be defined in the mongo configuration file, as well as allowing connections to the mongodb server from computers other than localhost:

2) Edit (maybe save a copy of the original file is a good idea before changing it) /etc/mongodb.conf (sudo <editor> /etc/mongodb.conf) for defining the replica set name and commenting out the bind_ip line:

```
==================
# replica name...
replSet = myreplica
…
#bind_ip = 127.0.0.1
…
==================
```

And that's all what we need to start the mongodb servers at each one of the computers:
3) $ sudo service mongodb start

3.1 Script for Starting with a Replica Set Configuration

Given that the replica set environment will be used for many different experiments/scenarios/tests, we will make a simple script for starting the mongodb with a replica set. The step-by-step guide is as follows.

1) Copy the original mongo configuration file in order to save a copy
    $ sudo cp /etc/mongdb.conf /etc/mongodb.orig

2) Prepare a configuration file to be specifically used for the mongo replication set
    $ sudo cp /etc/mongdb.conf /etc/mongodb.repl
And edit the replica set configuration file as described earlier (sudo <editor> /etc/mongodb.repl) for defining the replica set name and commenting out the bind_ip line:

```
==================
# replica name...
replSet = myreplica
…
#bind_ip = 127.0.0.1
…
==================
```

3) Create the script for restart the mongo operation with the replica set, we will call such script as “repl”:
repl
==========
sudo cp /etc/mongodb.conf.repl /etc/mongodb.conf
sudo service mongodb restart
==========

As a result of steps 1) and 2) above, there will be three mongo configuration files:
/etc/mongodb.conf (the one to be taken into account in mongo startup)
/etc/mongofb.conf.original (the one created at mongo install, with no replicaton set whatsoever)
/etc/mongodb.conf.repl (the one created for mongo replication set startup)

3.2 Starting the Replica Set Operation

Once every mongodb process (mongod) starts with a configured replica set, they are all peers and they are ready to start a NoSQL database replication, i.e.:
a) Vote a primary node among those in the same replica set.
b) Recognize non-primary node/s in the same replica set as secondary node/s.
c) Every change made in the primary node is replicated in every secondary node, as shown in Fig. 1.

From now on, the nodes are not all peers in the sense that not all the nodes are able to carry out the same operations for starting the replica set operation. More specifically, only one node should run the “initiate” operation:
1) $mongo
> rs.initiate()
….
> rs.status()
where rs.status() reports some details of the replication set currently operating. Even when the replication set operation has begun in the node in which rs.initiate() has been run, there is no actual replication because there is only one node in the replication set so far. The node in which the rs.initiate() has been successfully run has to include/reference the other nodes started with the same replication set in its configuration file. The rs.add() has to be used in so far unique node of the operating replication set. More specifically, if rs.initiate() has been run in the mv with IP 192.168.0.160, the nodes with IP 192.168.0.161 and 192.168.0.162 can be included/started in the replication set with:

2) > rs.add(‘192.168.0.161:27017’)
…
> rs.add(‘192.168.0.162:27017’)
…

The most frequent case is that the node in which the rs.add() is/are executed becomes the replication set primary node, while the others become secondary ones. However, there is no documentation on which one becomes the first primary node, only that an election is carried out when a replica set starts execution [3]. After adding nodes to the replica set, it is suggested to run rs.status() in order to show several details such as which node is the current primary node, and if every replication set node is recognized as “up & running”.

3.3 Script for Restart Operation without a Replica Set Configuration

We suggest a method to restart the entire system (the three vm) without replica set after having run the replica set system for a number of reasons:

- Restart the system from scratch for the cases in which some error in the configuration files has been included. Other problems arise in the case of executing rs.initiate() in more than one replication set node.
- Start the replica set in different scenarios, or at least in different nodes in order to test, for example, which node is elected as the first primary node (in the very first election).
- Performance analysis depending on different first-elected primary node.

The step-by-step guide is as follows.

1) Exit console (exit)

2) $ sudo service mongodb stop

3) Edit the mongodb configuration file so that the replica set is not started, comment out the line
   #replSet = myreplica
   in file /etc/mongodb.conf

4) $ sudo service mongodb start

5) $ mongo (warning about replica set problems…)
   > use local
   > db.dropDatabase()
   > exit

6) $ sudo service mongodb restart

If the mongo console is started, there is no more warning/indication of replica set problem/operation. We are able to automate the previous task with the following script, “norepl” which takes advantage of the configuration files already copied/generated in Section 3.1 above:
/etc/mongodb.conf (the one to be taken into account in mongo startup)
/etc/mongofb.conf.original (the one created at mongo install, with no replication set whatsoever)
/etc/mongodb.conf.repl (the one created for mongo replication set startup)

norepl

```
==========
sudo cp /etc/mongodb.conf.original /etc/mongodb.conf
sudo service mongodb restart
echo "Step 1:"
echo "  mongo"
echo "  > use local"
echo "  > db.dropDatabase()"
echo "  > exit"
echo ""
echo "Step 2:"
echo "  nothing"
==========
```

The script (norepl) should be run twice in all the nodes in order to get a mongo console with no replication set started in any mv.

4.- Conclusions and Further Work

We have presented a simple and straightforward step-by-step guide for having a replica set "up & running", along with a script to start every node in a replica set. We have automated the replica set start and, also, the restart of every node so that no replica set remains in operation. We have added some comments we have seen by experience and not fully documented in other tutorials such as [4] or is widespread in tutorials and reference manual/s.

We have not covered all alternatives, mostly because either other ways of configuring and starting the replica set are well documented or because the guide and comments in this guide let to better understand other alternative ways of starting the system. Clearly, configuration and initial operation of a mongo replica set is not useful by itself, we expect to run a number of tests and experiments for functionality and performance analysis.

References

[1] MongoDB, Inc, Replication, Replication - MongoDB Manual 3.4, 2017, https://docs.mongodb.com/manual/replication/

[2] MongoDB, Inc, Replica Set Members - MongoDB Manual 3.4, 2017 https://docs.mongodb.com/manual/core/replica-set-members/

[3] MongoDB, Inc, Replica Set Elections - MongoDB Manual 3.4, 2017, https://docs.mongodb.com/manual/core/replica-set-elections/

[4] MongoDB, Inc, Deploy a Replica Set - MongoDB Manual 3.4, 2017, https://docs.mongodb.com/manual/tutorial/deploy-replica-set/