Install and Use mongodb nodejs Driver with a mongdb Replica Set

Fernando G. Tinetti*, Agustín Terruzzi

Technical Report TR-BDD-02-2017
III-LIDI, Fac. de Informática, UNLP
*Also with CIC, Prov. de Buenos Aires
Argentina
contact e-mail: fernando@info.unlp.edu.ar
April 2017

Abstract. We document the installation of nodejs and the nodejs driver for mongodb operation from scratch, along with a few javascript scripts operating with a mongodb replica set. Even when there are many similar reports on nodejs and mongodb nodejs driver, we did not find anyone simple enough for a simple environment: nodejs code operating with an already installed mongodb replica set in a cluster-like environment. On one hand, some guides are too similar to a reference manual in that many details are assumed already known and only describe some commands and tools. On the other hand, there are many tutorials describing much more complex environment/s, including infrastructures/*frameworks* for web applications development and deployment, schema-like operations on mongodb, etc., none of which we are interested in, at least initially. We document in this report a step-by-step guide starting with a stand-alone computer (initially in the same LAN) without any infrastructure software other than the network connection and ending that computer being able to operate on a mongodb replica set database using javascript code. The initial goal is to have an environment for experimentation with a replicated mongodb database.

1.- Introduction

We are interested in having a nodejs application operating with a mongo replica set already installed and in operation in a cluster-like environment. Actually, the current configuration is described in Fig. 1: a computer with VirtualBox hosting three virtual machines (vm) on which a mongo replica set is running [1]. Even when the scenario in Fig. 1 is rather limited, it is not far from a general-purpose one. More

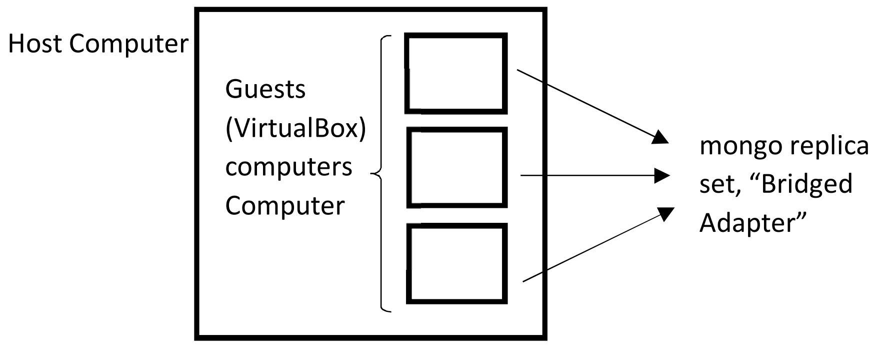

Host Computer
Guests (VirtualBox) computers Computer
mongo replica set, “Bridged Adapter”

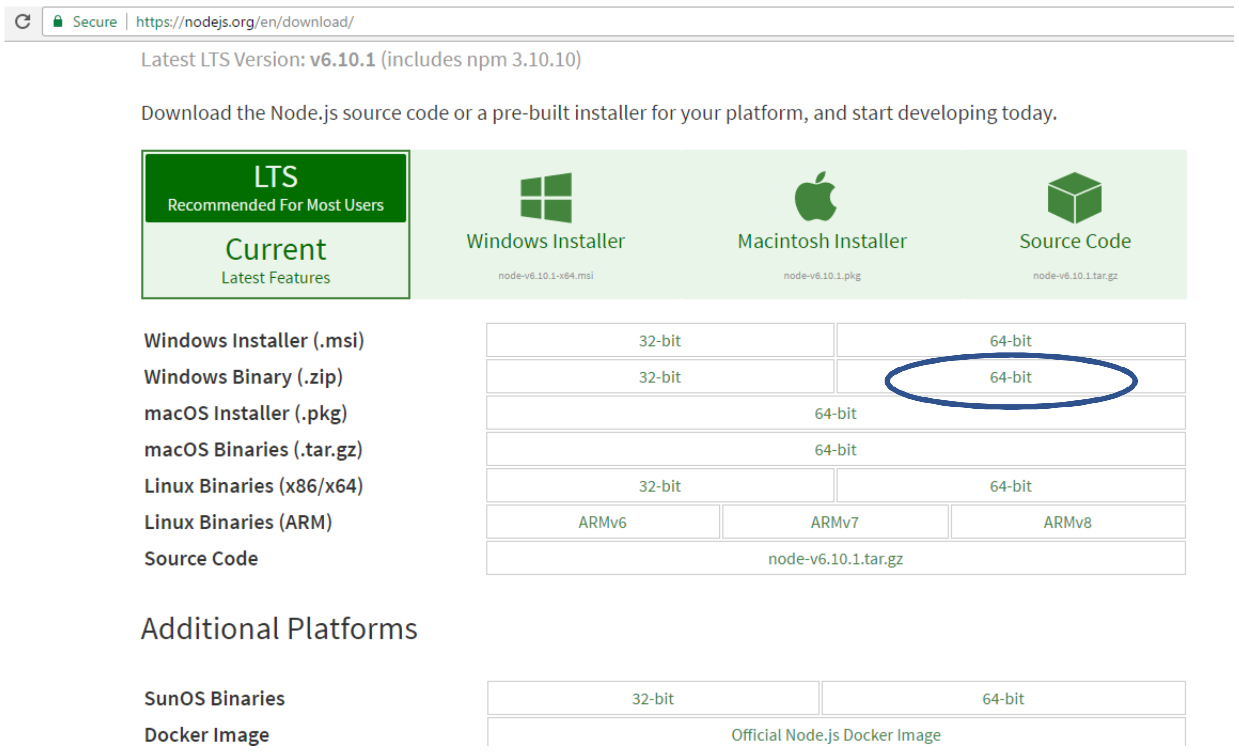
Figure 1: Initial Environment.

specifically, we replicate the cluster-like environment in which the mongo replica set usually run. The relationship/coupling among VirtualHost host-guests is clearly far from being a production environment, but allows a single environment for carrying out several experiments in an affordable infrastructure. We specifically simplified the network setting/s to the VirtualBox “Bridged Adapter” so that the vm have internet access by default and the host computer is in the same LAN as the vm.

In this report, we make a step-by-step guide for the installation, configuration, and usage of nodejs and the nodejs mongodb driver in a computer so that a nodejs application is able to operate on a mongodb replica set. Since the host computer we are using has a windows operating system (OS), the installation details will be “restricted” to that OS, but the general configuration and nodejs code is, in fact, platform independent, so we expect this report to be useful in several other scenarios as well. As a “collateral effect” of installing nodejs and the mongodb nodejs driver in a Windows computer, we will have a heterogeneous environment, which is rather usual in the context of distributed systems.

2.- Installation and Configuration of nodejs

The nodejs software infrastructure is obtained at its web site [2], as shown in Fig. 2 for the Windows OS without installer (aka binary distribution).



Secure | https://nodejs.org/en/download/

Latest LTS Version: **v6.10.1** (includes npm 3.10.10)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

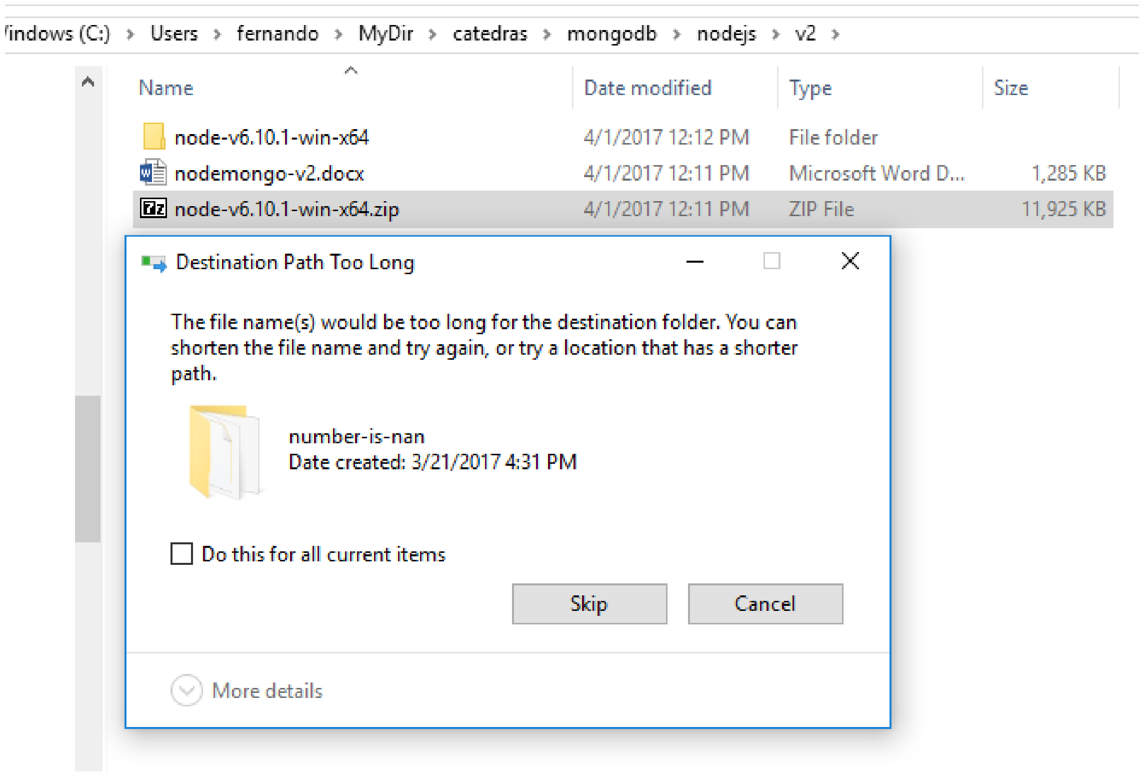
Windows Installer (.msi)	32-bit		64-bit
Windows Binary (.zip)	32-bit		[✓ 64-bit]
macOS Installer (.pkg)	64-bit		
macOS Binaries (.tar.gz)	64-bit		
Linux Binaries (x86/x64)	32-bit		64-bit
Linux Binaries (ARM)	ARMv6	ARMv7	ARMv8
Source Code	node-v6.10.1.tar.gz		

Additional Platforms

SunOS Binaries	32-bit	64-bit
Docker Image	Official Node.js Docker Image	

Figure 2: Download Windows Binary nodejs.

Trying to install in the selected place (c:\users\fernando\mydir\catedras\mongodb\nodejs\v2) generates an error, as shown in Fig. 3. The error may be due to the program used to handle the .zip, 7-zip.



'indows (C:) › Users › fernando › MyDir › catedras › mongodb › nodejs › v2 ›

Name	Date modified	Type	Size
node-v6.10.1-win-x64	4/1/2017 12:12 PM	File folder	
nodemongo-v2.docx	4/1/2017 12:11 PM	Microsoft Word D...	1,285 KB
node-v6.10.1-win-x64.zip	4/1/2017 12:11 PM	ZIP File	11,925 KB

Destination Path Too Long

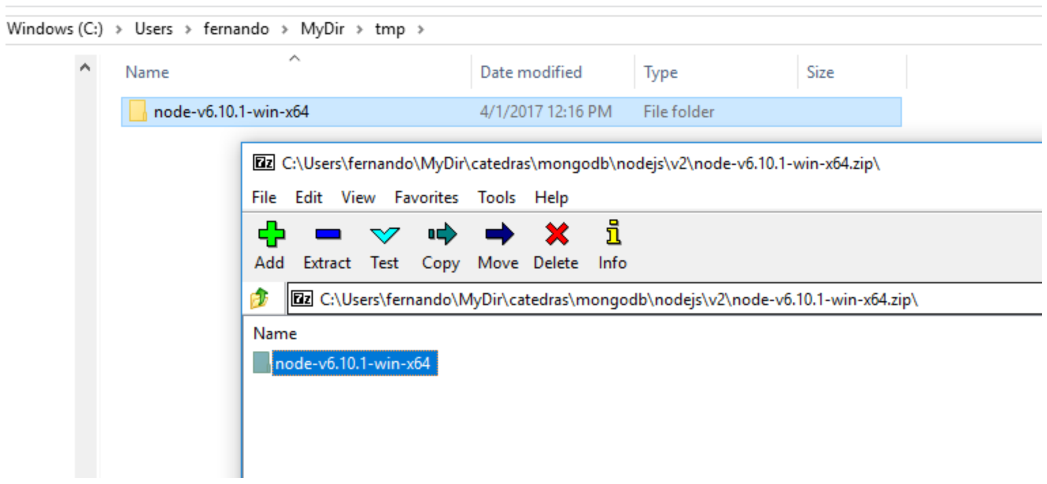
The file name(s) would be too long for the destination folder. You can shorten the file name and try again, or try a location that has a shorter path.

number-is-nan
Date created: 3/21/2017 4:31 PM

Do this for all current items

Figure 3: Problem in the Binary Unzip and Copy.

We are aware this is a minor error and unrelated to nodejs, but we want to document it in either case, just as an example of minor details/problems found in this kind of nodejs installation. The work-around to this problem is simple: unzip to a "short" pathname directory (c:\users\fernando\mydir\tmp) and copy de unzipped material to the original/selected one, as shown in Fig. 4 and Fig. 5.



Windows (C:) › Users › fernando › MyDir › tmp ›

Name	Date modified	Type	Size
node-v6.10.1-win-x64	4/1/2017 12:16 PM	File folder	

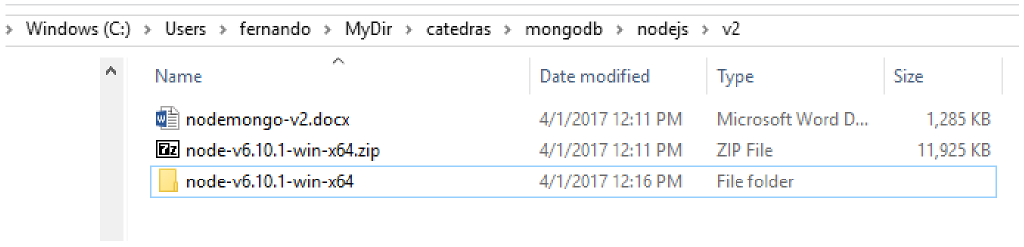
C:\Users\fernando\MyDir\catedras\mongodb\nodejs\v2\node-v6.10.1-win-x64.zip\

File Edit View Favorites Tools Help
Add
Extract
Test
Copy
Move
Delete
Info

C:\Users\fernando\MyDir\catedras\mongodb\nodejs\v2\node-v6.10.1-win-x64.zip\

Name
node-v6.10.1-win-x64

Figure 4: Unzip to a "Short" Pathname Directory.



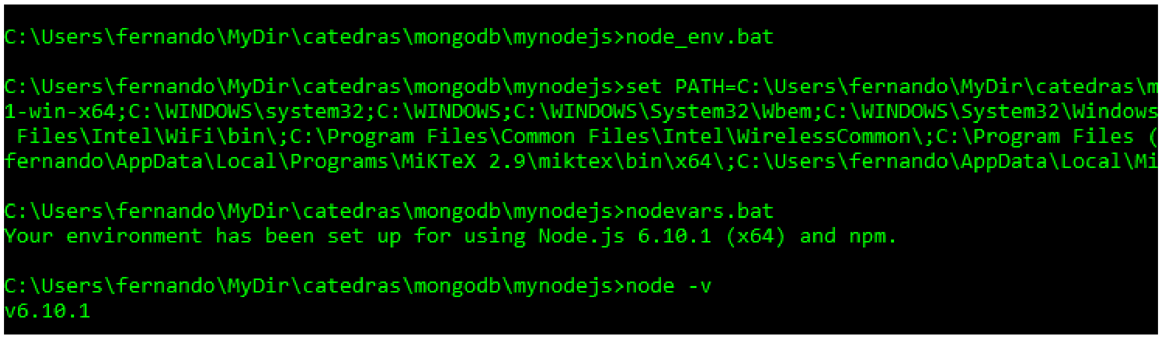
```
> Windows (C:) > Users > fernando > MyDir > catedras > mongodb > nodejs > v2
Name                        Date modified       Type                  Size
nodemongo-v2.docx           4/1/2017 12:11 PM   Microsoft Word D...   1,285 KB
node-v6.10.1-win-x64.zip    4/1/2017 12:11 PM   ZIP File              11,925 KB
node-v6.10.1-win-x64        4/1/2017 12:16 PM   File folder
```

Figure 5: nodejs Directory.

Given that we chose to install binaries, we have to set path and other environment variables. We have defined a node_env.bat file to do so:

```
Node_env.bat
=================================================================
set PATH=C:\Users\fernando\MyDir\catedras\mongodb\nodejs\v2\node-v6.10.1-win-x64;%PATH%
nodevars.bat
=================================================================
```

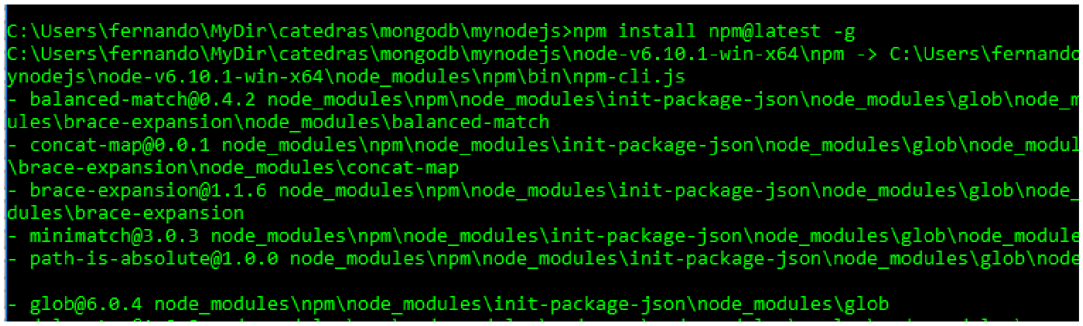
The installation or, at least, the node executable functionality can be verified with the command
node -v
as shown in Fig. 6, where the node command shows the node version installed in the computer.



```
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>node_env.bat

C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>set PATH=C:\Users\fernando\MyDir\catedras\m
1-win-x64;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\Windows
 Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\;C:\Program Files (
fernando\AppData\Local\Programs\MiKTeX 2.9\miktex\bin\x64\;C:\Users\fernando\AppData\Local\Mi

C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>nodevars.bat
Your environment has been set up for using Node.js 6.10.1 (x64) and npm.

C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>node -v
v6.10.1
```

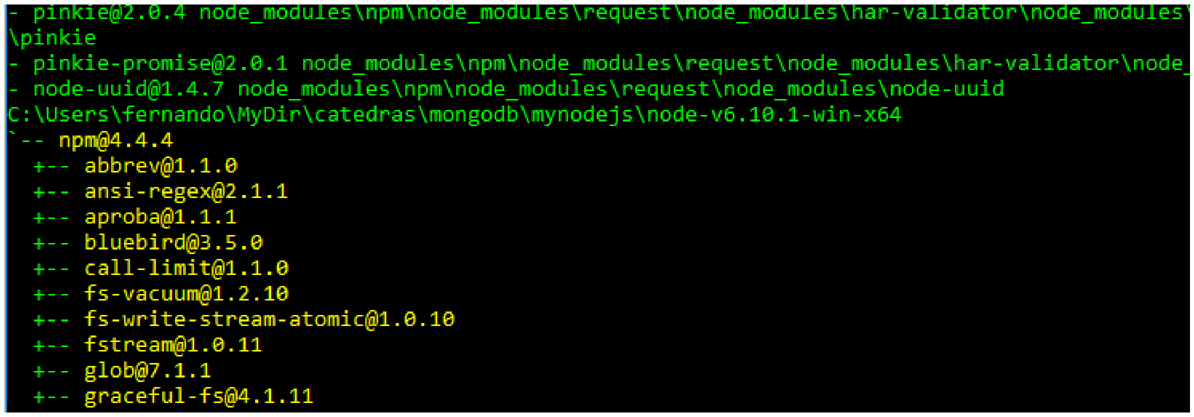
Figure 5: nodejs Installation Minimum Check.

And install the npm "environment", as suggested in [3] [4] [5], because most nodejs application will depend on several nodejs modules, easily managed by npm. The npm installation is made with
npm install npm@latest -g
as shown in Fig. 6, where it is also shown some of the actual installation process output.



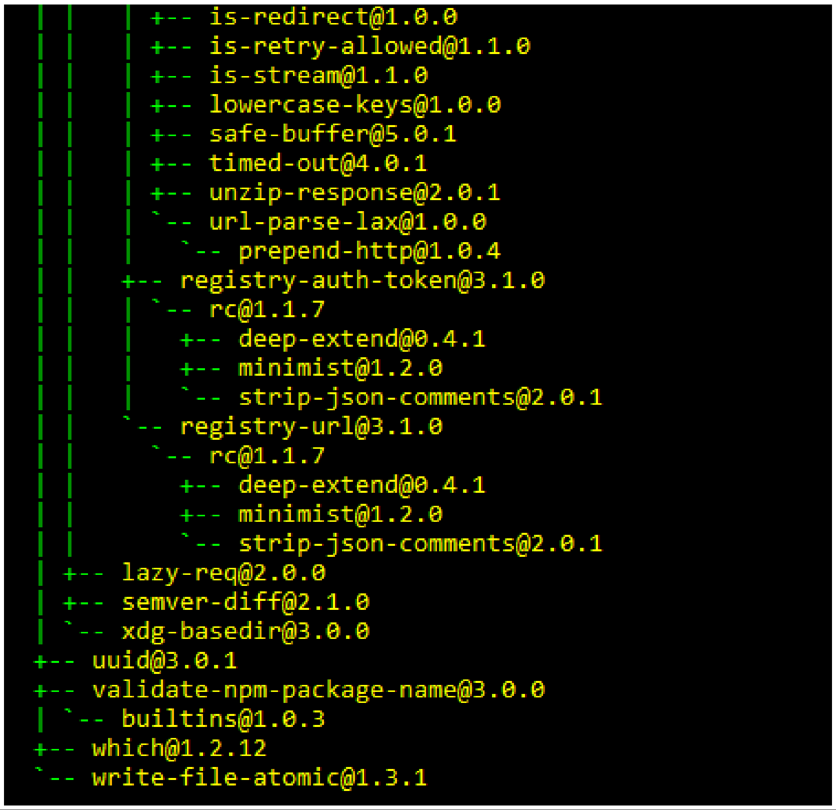
```
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>npm install npm@latest -g
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node-v6.10.1-win-x64\npm -> C:\Users\fernando
ynodejs\node-v6.10.1-win-x64\node_modules\npm\bin\npm-cli.js
- balanced-match@0.4.2 node_modules\npm\node_modules\init-package-json\node_modules\glob\node_m
ules\brace-expansion\node_modules\balanced-match
- concat-map@0.0.1 node_modules\npm\node_modules\init-package-json\node_modules\glob\node_modul
\brace-expansion\node_modules\concat-map
- brace-expansion@1.1.6 node_modules\npm\node_modules\init-package-json\node_modules\glob\node_
dules\brace-expansion
- minimatch@3.0.3 node_modules\npm\node_modules\init-package-json\node_modules\glob\node_module
- path-is-absolute@1.0.0 node_modules\npm\node_modules\init-package-json\node_modules\glob\node

- glob@6.0.4 node_modules\npm\node_modules\init-package-json\node_modules\glob
```

… and lots of similar output, up to



```
- pinkie@2.0.4 node_modules\npm\node_modules\request\node_modules\har-validator\node_modules
\pinkie
- pinkie-promise@2.0.1 node_modules\npm\node_modules\request\node_modules\har-validator\node_
- node-uuid@1.4.7 node_modules\npm\node_modules\request\node_modules\node-uuid
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node-v6.10.1-win-x64
`-- npm@4.4.4
  +-- abbrev@1.1.0
  +-- ansi-regex@2.1.1
  +-- aproba@1.1.1
  +-- bluebird@3.5.0
  +-- call-limit@1.1.0
  +-- fs-vacuum@1.2.10
  +-- fs-write-stream-atomic@1.0.10
  +-- fstream@1.0.11
  +-- glob@7.1.1
  +-- graceful-fs@4.1.11
```

… and lots of similar output, up to

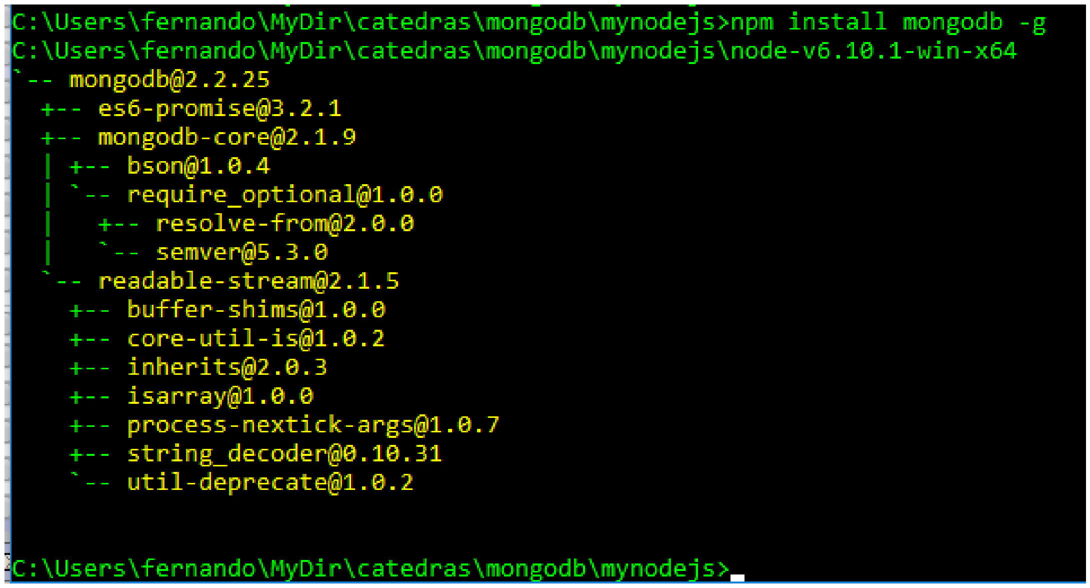


```
  | |   | +-- is-redirect@1.0.0
  | |   | +-- is-retry-allowed@1.1.0
  | |   | +-- is-stream@1.1.0
  | |   | +-- lowercase-keys@1.0.0
  | |   | +-- safe-buffer@5.0.1
  | |   | +-- timed-out@4.0.1
  | |   | +-- unzip-response@2.0.1
  | |   | `-- url-parse-lax@1.0.0
  | |   |   `-- prepend-http@1.0.4
  | |   +-- registry-auth-token@3.1.0
  | |   | `-- rc@1.1.7
  | |   |   +-- deep-extend@0.4.1
  | |   |   +-- minimist@1.2.0
  | |   |   `-- strip-json-comments@2.0.1
  | |   `-- registry-url@3.1.0
  | |     `-- rc@1.1.7
  | |       +-- deep-extend@0.4.1
  | |       +-- minimist@1.2.0
  | |       `-- strip-json-comments@2.0.1
  | +-- lazy-req@2.0.0
  | +-- semver-diff@2.1.0
  | `-- xdg-basedir@3.0.0
  +-- uuid@3.0.1
  +-- validate-npm-package-name@3.0.0
  | `-- builtins@1.0.3
  +-- which@1.2.12
  `-- write-file-atomic@1.3.1
```

Figure 6: Installation/Update of npm.

3.- Installation and First Tests of the mongodb nodejs Driver

The nodejs driver installation is straightforward [6], according to “Install MongoDB Node.js Driver”
C:\Users\fernando>npm install mongodb -g
(-g option used just in case install node module as global module…) as shown in Fig. 7.



```
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>npm install mongodb -g
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node-v6.10.1-win-x64
`-- mongodb@2.2.25
  +-- es6-promise@3.2.1
  +-- mongodb-core@2.1.9
  | +-- bson@1.0.4
  | `-- require_optional@1.0.0
  |   +-- resolve-from@2.0.0
  |   `-- semver@5.3.0
  `-- readable-stream@2.1.5
    +-- buffer-shims@1.0.0
    +-- core-util-is@1.0.2
    +-- inherits@2.0.3
    +-- isarray@1.0.0
    +-- process-nextick-args@1.0.7
    +-- string_decoder@0.10.31
    `-- util-deprecate@1.0.2


C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>
```

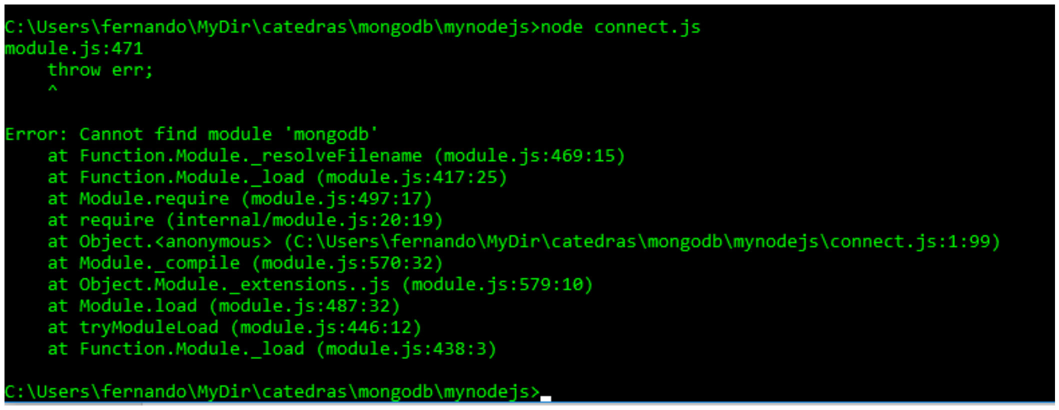
Figure 7: mongodb nodejs Driver Install.

At this point, a nodejs application is able to connect to a mongodb database, including one handled by a replica set given that the connection is made to the server acting as the replica set primary. Following the guide at [6], the code in connect.js containing
connect.js

```
==================================================================
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');

var url = 'mongodb://192.168.0.160:27017/test';
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected correctly to server.");
  db.close();
});
==================================================================
```

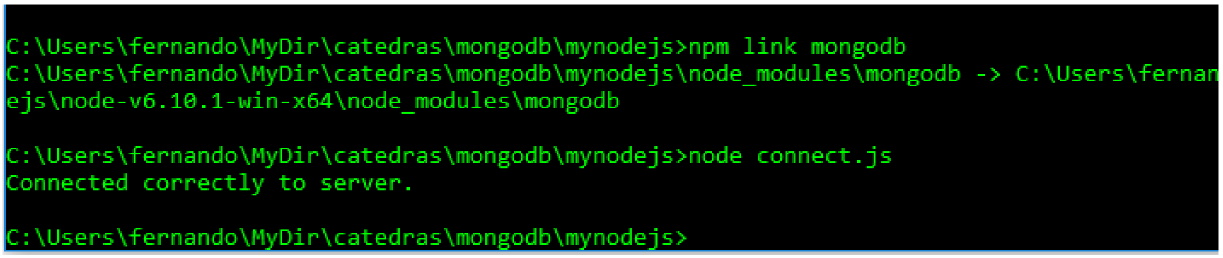
(the mongo replica set primary node is running at 192.168.0.160, and listens to connections in the default mongodb server port) is used as an example for database connection as shown in Fig. 8 with
node connect.js

where an error is reported by nodejs (as expected, actually). The problem is not related to mongo, but to the nodejs module handling: connect.js requires the module “mongodb”, which is already installed as a global module, but not “reachable” by the connect.js execution environment (yet). More specifically,
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>node connect.js
Fails because the mongodb module is not directly related to this nodejs code. The mongodb module has to be “linked” to this code with
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>npm link mongodb
Thus, the right sequence would be (the first step is needed only once)
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>npm link mongodb
…
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>node connect.js
as shown in Fig. 9, where the successful output is also shown, i.e.
C:\Users\fernando\MyDir\catedras\mongodb\nodejs>node connect.js
Connected correctly to server.



```
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>node connect.js
module.js:471
    throw err;
    ^

Error: Cannot find module 'mongodb'
    at Function.Module._resolveFilename (module.js:469:15)
    at Function.Module._load (module.js:417:25)
    at Module.require (module.js:497:17)
    at require (internal/module.js:20:19)
    at Object.<anonymous> (C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\connect.js:1:99)
    at Module._compile (module.js:570:32)
    at Object.Module._extensions..js (module.js:579:10)
    at Module.load (module.js:487:32)
    at tryModuleLoad (module.js:446:12)
    at Function.Module._load (module.js:438:3)

C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>
```

Figure 8: First Attempt to mongodb Connection.



```
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>npm link mongodb
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node_modules\mongodb -> C:\Users\fernan
ejs\node-v6.10.1-win-x64\node_modules\mongodb

C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>node connect.js
Connected correctly to server.

C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>
```

Figure 9: Link mongodb Driver Module and run mongodb Connection.

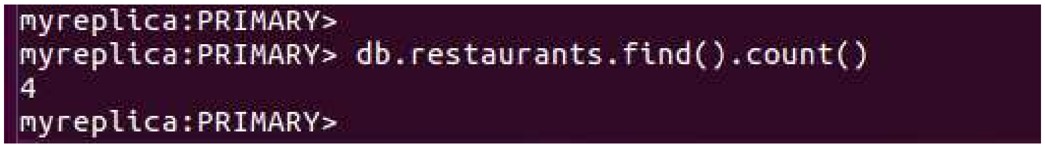
It is worth noting that the connection is possible given that the computer where the javascript code is run is able to reach the mongodb replica set primary node. More specifically, this is one of the reasons why the VirtualBox host computer is in the same LAN as the vm with the mongodb replica set, given that every vm network is defined as a so called “Bridged Adapter” in the VirtualBox vm manager.

The guide at [7] is followed in order to insert documents in a mongodb database, i.e. the insert.js code connects to a mongodb and inserts a document, where insert.js is as follows (directly copied from [7] in order to avoid confusions for changing the code).

insert.js

```
==========================================================
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');
var url = 'mongodb://192.168.0.160:27017/test';

var insertDocument = function(db, callback) {
  db.collection('restaurants').insertOne( {
    "address" : {
      "street" : "2 Avenue",
      "zipcode" : "10075",
      "building" : "1480",
      "coord" : [ -73.9557413, 40.7720266 ]
    },
    "borough" : "Manhattan",
    "cuisine" : "Italian",
    "grades" : [
      {
        "date" : new Date("2014-10-01T00:00:00Z"),
        "grade" : "A",
        "score" : 11
      },
      {
        "date" : new Date("2014-01-16T00:00:00Z"),
        "grade" : "B",
        "score" : 17
      }
    ],
    "name" : "Vella",
    "restaurant_id" : "41704620"
  }, function(err, result) {
   assert.equal(err, null);
   console.log("Inserted a document into the restaurants collection.");
   callback();
 });
};

MongoClient.connect(url, function(err, db) {
 assert.equal(null, err);
 insertDocument(db, function() {
     db.close();
 });
})
==========================================================
```

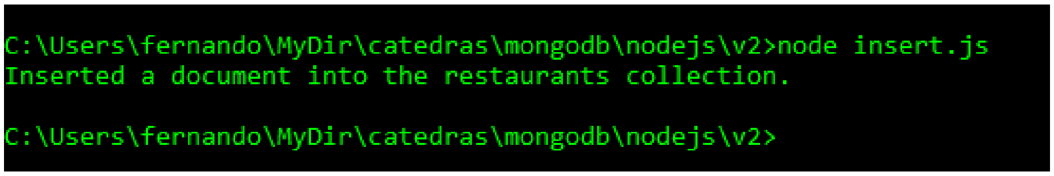
Before using insert.js, the collection is such that it has already four documents (due to previous experiments), as shown in Fig. 9, using the mongo console in the mongodb replica set primary node.



```
myreplica:PRIMARY>
myreplica:PRIMARY> db.restaurants.find().count()
4
myreplica:PRIMARY>
```

Figure 9: Number of Documents Before Using insert.js.

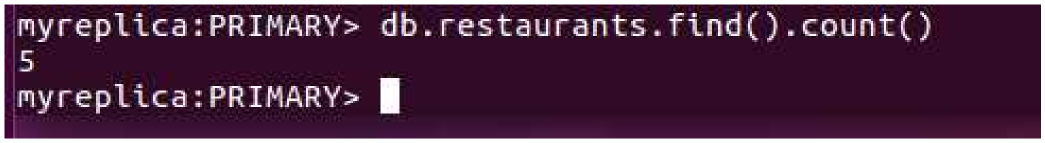
And using insert.js (which connects to the mongodb replica set primary node, running at 192.168.0.160) as shown in Fig. 10 the result is reported successful in the client side (i.e. where the javascript code runs).



```
C:\Users\fernando\MyDir\catedras\mongodb\nodejs\v2>node insert.js
Inserted a document into the restaurants collection.

C:\Users\fernando\MyDir\catedras\mongodb\nodejs\v2>
```

Figure 10: Insert a Document with insert.js.

In the server side, after using insert.js, the collection is such that there is one more document, as expected after inserting one, as shown in Fig. 11.



```
myreplica:PRIMARY> db.restaurants.find().count()
5
myreplica:PRIMARY>
```

Figure 11: Number of Documents after Using insert.js.

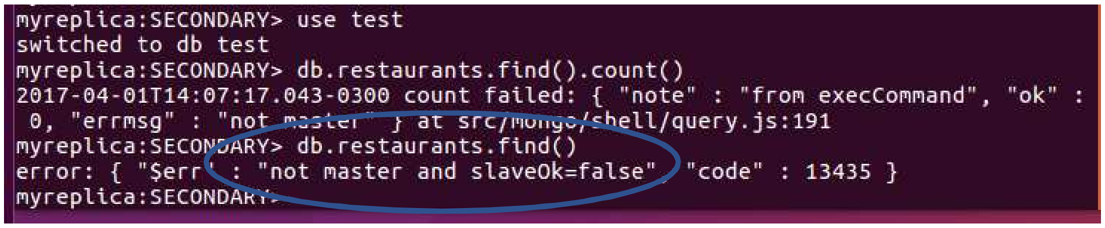
As explained above, the javascript client will be able to connect and operate on the mongodb database replica set only if it uses the mongodb replica set primary node. One of the main disadvantages of this type of clients is that the server working as the replica set primary node is defined at runtime. In our experiments, the node in which the replica set is "ininiate()d" becomes the primary node, but it would change at runtime depending on runtime conditions (e.g. due to node reset/shutdown, and/or network errors). If the server explicitly referenced in the .js files is shutdown, the nodejs code reports error/s (font has been reduced in order to void too many text line wraparounds):

```
C:\Users\fernando\MyDir\catedras\mongodb\nodejs>node connect.js

C:\Users\fernando\MyDir\catedras\mongodb\nodejs\node_modules\mongodb\lib\mongo_client.js:338
        throw err
        ^
AssertionError: null == { MongoError: failed to connect to server [192.168.0.160:27017] on first connect
[MongoError: connect ETIMEDOUT 192.168.0.160:27
    at C:\Users\fernando\MyDir\catedras\mongodb\nodejs\connect.js:6:10
    at connectCallback
(C:\Users\fernando\MyDir\catedras\mongodb\nodejs\node_modules\mongodb\lib\mongo_client.js:428:5)
    at C:\Users\fernando\MyDir\catedras\mongodb\nodejs\node_modules\mongodb\lib\mongo_client.js:335:11
    at _combinedTickCallback (internal/process/next_tick.js:73:7)
    at process._tickCallback (internal/process/next_tick.js:104:9)
```

Otherwise, if the server explicitly referenced in the .js file is not the replica set primary node, the nodejs code reports error/s too:

```
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>node insert.js

C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node-v6.10.1-win-x64\node_modules\mongodb\lib\utils.js:123
    process.nextTick(function() { throw err; });
                                  ^
AssertionError: { MongoError: not master
    at Function.MongoError.create (C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node-v6.10.1-win-x == null
    at C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\insert.js:30:12
    at C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node-v6.10.1-win-
x64\node_modules\mongodb\lib\collection.js:431:32
    at handleCallback (C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node-v6.10.1-win-
x64\node_modules\mongodb\lib\utils.js:120:56)
    at C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node-v6.10.1-win-
x64\node_modules\mongodb\lib\collection.js:736:20
    at C:\Users\fernando\MyDir\catedras\mongodb\mynodejs\node-v6.10.1-win-
x64\node_modules\mongodb\node_modules\mongodb-core\lib\connection\pool.js:461:18
    at _combinedTickCallback (internal/process/next_tick.js:73:7)
    at process._tickCallback (internal/process/next_tick.js:104:9)
```

Actually, the second error reported at the nodejs client side is similar to that reported at the mongo console of the replica set secondary nodes, as shown in Fig. 12, where the error report indicates a possible solution for reading/querying the database on secondary nodes (slaveOk=true).



```
myreplica:SECONDARY> use test
switched to db test
myreplica:SECONDARY> db.restaurants.find().count()
2017-04-01T14:07:17.043-0300 count failed: { "note" : "from execCommand", "ok" :
 0, "errmsg" : "not master" } at src/mongo/shell/query.js:191
myreplica:SECONDARY> db.restaurants.find()
error: { "$err" : "not master and slaveOk=false", "code" : 13435 }
myreplica:SECONDARY>
```

Figure 12: Error at Replica Set Secondary Node Trying to Query the Database.

The description so far covers only basic operations (there are many more details, such as those in [8]) and it does not use/take advantage of a mongo replica set at the client side. The replica set provides high availability in case of some network and server errors and possible performance enhancement for reading operations. The given code examples do not take advantage of such high availability facilities because the connection is made with a single mongodb server, not with a replica set. The next section provides some simple examples and explanations on a nodejs client (and setting apart the javascript code, any other client) connecting to a replica set, not a single server.

4.- A nodejs Client of a mongodb Replica Set "Server"

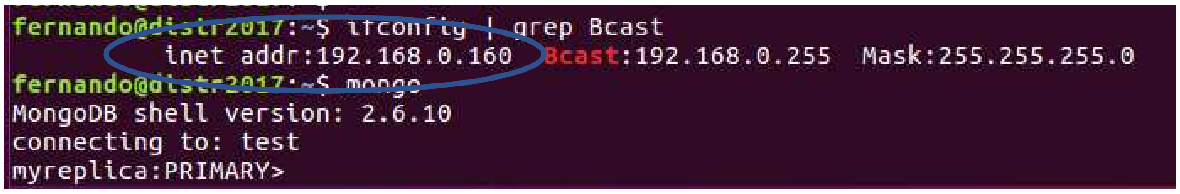
Connecting a nodejs application to a mongodb replica set [9] implies using a specific URI format [10]. The nodejs driver has specific options too, as described at [11], some of which are related to replica set connections. The minimum suggested information in the URI provided for the connection to a replica set would be the list of servers (referred to as "a seedlist of replica set members") and the replica set name. Taking into account we have a mongodb replica set "up & running" as described in [tecrepl], i.e.:

- Computers IPs 192.168.0.160, 192.168.0.161, and 192.168.0.162,
- Every mongodb server (mongod) using the server default port,
- The replica set name is “myreplica”,
- The databse is ‘test’

the URL we should use for the client connection should be
'mongodb://192.168.0.160,192.168.0.161,192.168.0.162/test?replicaSet=myreplica'
Thus, the “complete” javascript code for a client connection to a mongodb replica set, connectrepl.js, would be as follows
connectrepl.js

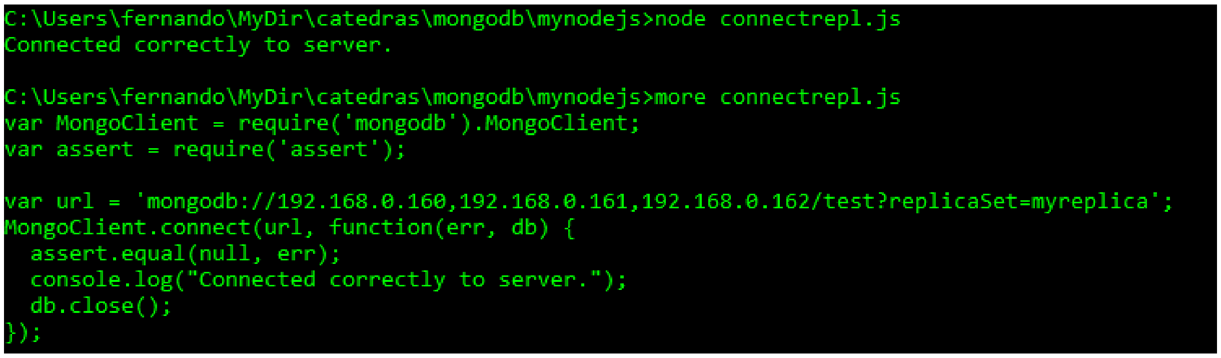
```
==========================================================================
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');

var url = 'mongodb://192.168.0.160,192.168.0.161,192.168.0.162/test?replicaSet=myreplica';
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected correctly to server.");
  db.close();
});
==========================================================================
```

which is identical to the one given as connect.js except for the url value (now referencing the replica whole set, including the replica set name). Take into account that the so called seedlist of replica set members is given without any identification of primary and/or secondary nodes, as expected, since it is not possible to know in advance which of the nodes will be the primary one. Fig. 13 shows that the node with IP 192.168.0.160 the current replica set primary node, and Fig. 14 shows that the connection is successful.



```
fernando@distr2017:~$ ifconfig | grep Bcast
          inet addr:192.168.0.160  Bcast:192.168.0.255  Mask:255.255.255.0
fernando@distr2017:~$ mongo
MongoDB shell version: 2.6.10
connecting to: test
myreplica:PRIMARY>
```

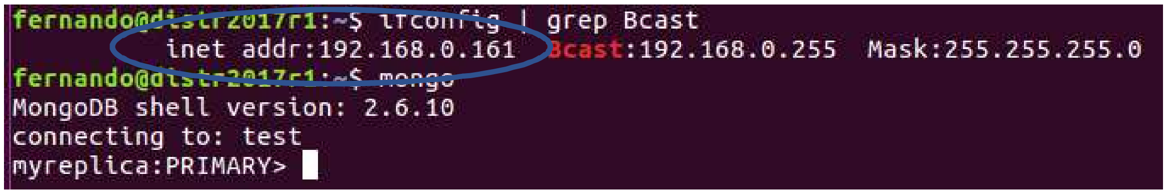
Figure 13: Replica Set Primary Node: 192.168.0.160.



```
C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>node connectrepl.js
Connected correctly to server.

C:\Users\fernando\MyDir\catedras\mongodb\mynodejs>more connectrepl.js
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');

var url = 'mongodb://192.168.0.160,192.168.0.161,192.168.0.162/test?replicaSet=myreplica';
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected correctly to server.");
  db.close();
});
```

Figure 14: connectrepl.js Successful Connection to the Replica Set.

Furthermore, Fig. 15 shows the scenario in which the node with IP 192.168.0.161 is the current replica set primary node, and the connectrepl.js is still successful, it connects to the replica set independently of the current replica set primary node.



```
fernando@distr2017r1:~$ ifconfig | grep Bcast
          inet addr:192.168.0.161  Bcast:192.168.0.255  Mask:255.255.255.0
fernando@distr2017r1:~$ mongo
MongoDB shell version: 2.6.10
connecting to: test
myreplica:PRIMARY>
```

Figure 15: Replica Set Primary Node: 192.168.0.161.

By default, the mongodb client operates with the replica set primary node, i.e. read and write database operations are made in the replica set primary node. Among the settings that can be changed, read operations can be made in replica set secondary nodes if required, given the proper connection setting/s option/s, e.g. Setting the readPreference value to one of

ReadPreference.PRIMARY
ReadPreference.PRIMARY_PREFERRED
ReadPreference.SECONDARY
ReadPreference.SECONDARY_PREFERRED
ReadPreference.NEAREST

(given as a MongoClient.connect function option [11]).

5.- Conclusions and Further Work

We have set up a nodejs execution environment, installed the mongodb driver so that it is possible to connect to a mongodb database, and show how to connect to a mongodb replica set in particular (i.e. a replicated mongodb database). The nodejs installation and project management mostly through nom has a lot of details not covered in this step-by-step guide. Instead, we have restricted to the simplest environment/ installation in a Windows computer. We think each nodejs developer has a predefined development environment and it would be almost impossible to cover them all.

We have used a mongodb replica set already installed so that it was possible to show how to make connections and how the client is transparently managed by the mongodb replica set independently of the details of the current replica set primary node. We know this is just the minimum detail covered in this report regarding mongodb replica set databases. Actually, this step-by-step guide allows to have an experimentation environment we think would be very useful to analyze at least mongodb high availability facilities to clients (nodejs clients in this case). In this context, the guide we presented is only about infrastructure, the interesting details will be obtained by thoroughly thought through experiments that should provide insight on specific performance details.

References

[1] Fernando G. Tinetti, Agustín Terruzzi, "Install a mongodb Replica Set: 1 Primary Node and 2 Secondary Nodes", Technical Report BDD-01-2017, March 2017.

http://fernando.bl.ee/reptec/repl-mongo.pdf
[2] 2017 Node.js Foundation, "Download | Node,js",
https://nodejs.org/en/download/
[3] Tierney Cyren, Installing Node.js Tutorial: Windows, 2017
https://nodesource.com/blog/installing-nodejs-tutorial-windows/
[4] NPM Inc., Get npm,
https://www.npmjs.com/get-npm
[5] Installing Node.js and updating npm,
https://docs.npmjs.com/getting-started/installing-node
[6] MongoDB, Inc., Node.js Driver - Getting Started With MongoDB 3.0.4,
https://docs.mongodb.com/getting-started/node/client/
[7] MongoDB, Inc., Insert Data with Node.js - Getting Started With MongoDB 3.0.4,
https://docs.mongodb.com/getting-started/node/insert/
[8] Tutorials, CRUD Operations,
http://mongodb.github.io/node-mongodb-native/2.2/tutorials/crud/
[9] Tutorials, Connect to MongoDB,
http://mongodb.github.io/node-mongodb-native/2.2/tutorials/connect/
[10] MongoDB, Inc., Connection String URI Format, MongoDB Manual 3.4,
https://docs.mongodb.com/manual/reference/connection-string/
[11] URI Connection Settings,
http://mongodb.github.io/node-mongodb-native/2.2/reference/connecting/connection-settings/