Data Durability in Replica Sets

In this lesson, we will look at what happens to the data, stored in a database when a secondary node is killed off.

WE'LL COVER THE FOLLOWING

Now that you are clear on how replication works and how it is implemented let's look into some other details.

In one of the previous lessons, what happens when a *primary node* fails.

However, what happens if one of the *secondary* nodes is killed?

We learned previously that the goal of replication is to ensure that no data is lost if a server fails. Hence, the data is still available even if a secondary node is killed off.

Implementation

Now, let's create a *replica set*, kill a *secondary* node, and then look at its impact on the data stored in the database.

Follow the steps below:

Follow the steps below.

Creating a replica set

As in the previous lesson, the three `mongod` instances are initiated first.

> The three commands, for running the `mongod` instances, have already been run for you in the terminal below.

To see the details of the commands used to run these `mongod` instances, go to the following lesson.

Starting the Mongo Shell

Next, connect to one of the running `mongod` processes using the MongoDB shell client.

Call `mongo` to connect to the `mongod` process that is running on port `27017`.

> **Note:** This will act as our *primary* node.

Type the command below on the terminal:

```
mongo
```

Running "mongo" to connect to port 27017

Using `rs.initiate()`

As you did previously, call the following command in the terminal next:

```
rs.initiate({_id : "rs0",members: [{_id : 0, host : "localhost:27017"}]})
```

Calling rs.initiate() command

At this point, your *replica* set has been initiated.

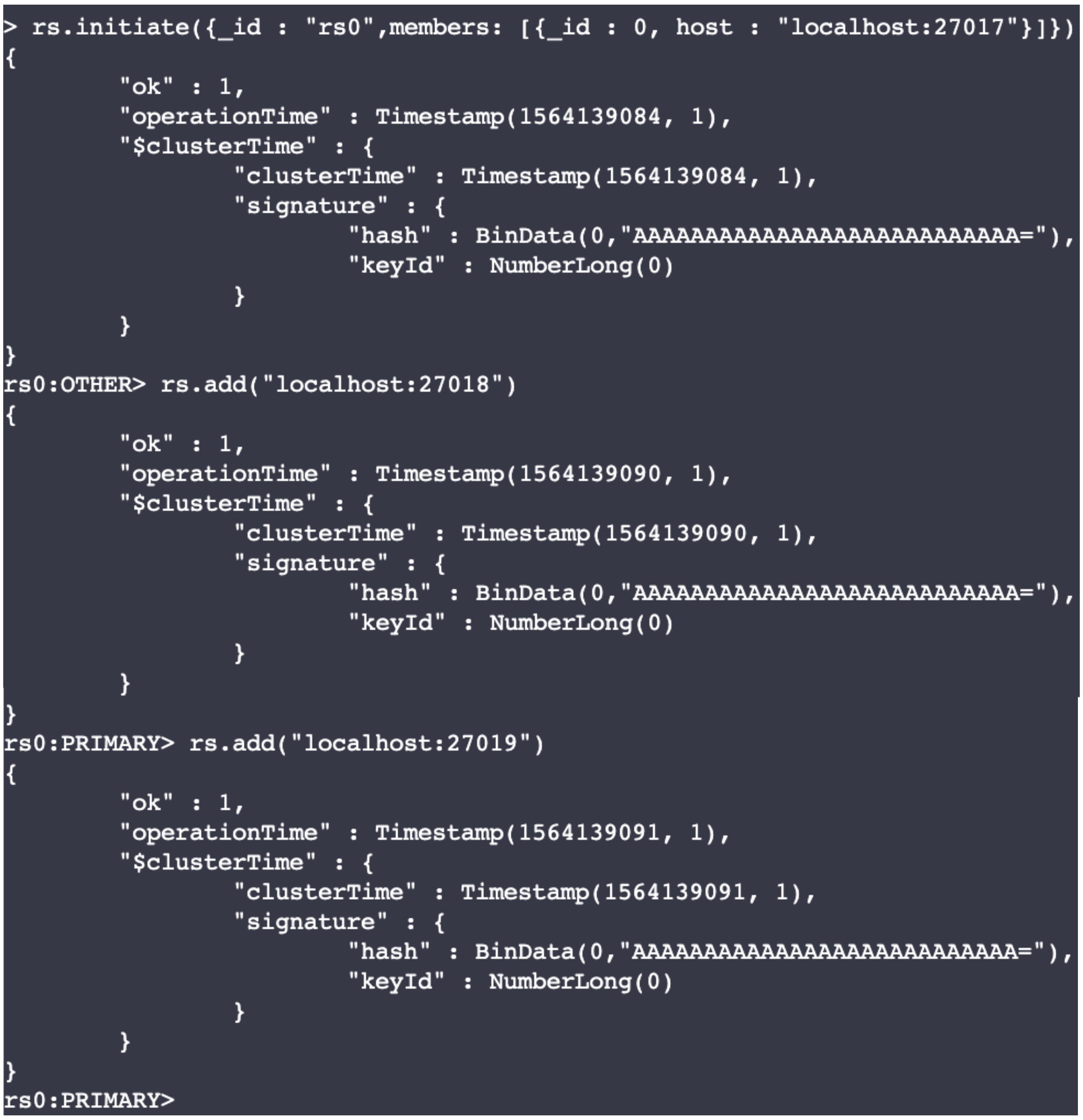
Using `rs.add()`

As done in the previous lesson, add the *secondary nodes* into `members` by typing the following commands in the terminal:

```
rs.add("localhost:27018")
rs.add("localhost:27019")
```

Running rs.add() commands for the other two ports

You should see something like this:



```
> rs.initiate({_id : "rs0",members: [{_id : 0, host : "localhost:27017"}]})
{
        "ok" : 1,
        "operationTime" : Timestamp(1564139084, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1564139084, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rs0:OTHER> rs.add("localhost:27018")
{
        "ok" : 1,
        "operationTime" : Timestamp(1564139090, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1564139090, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rs0:PRIMARY> rs.add("localhost:27019")
{
        "ok" : 1,
        "operationTime" : Timestamp(1564139091, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1564139091, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rs0:PRIMARY>
```

Setting up a Database

Next, set up a database with a collection called `users` (refer to the following lesson to learn how to do that).

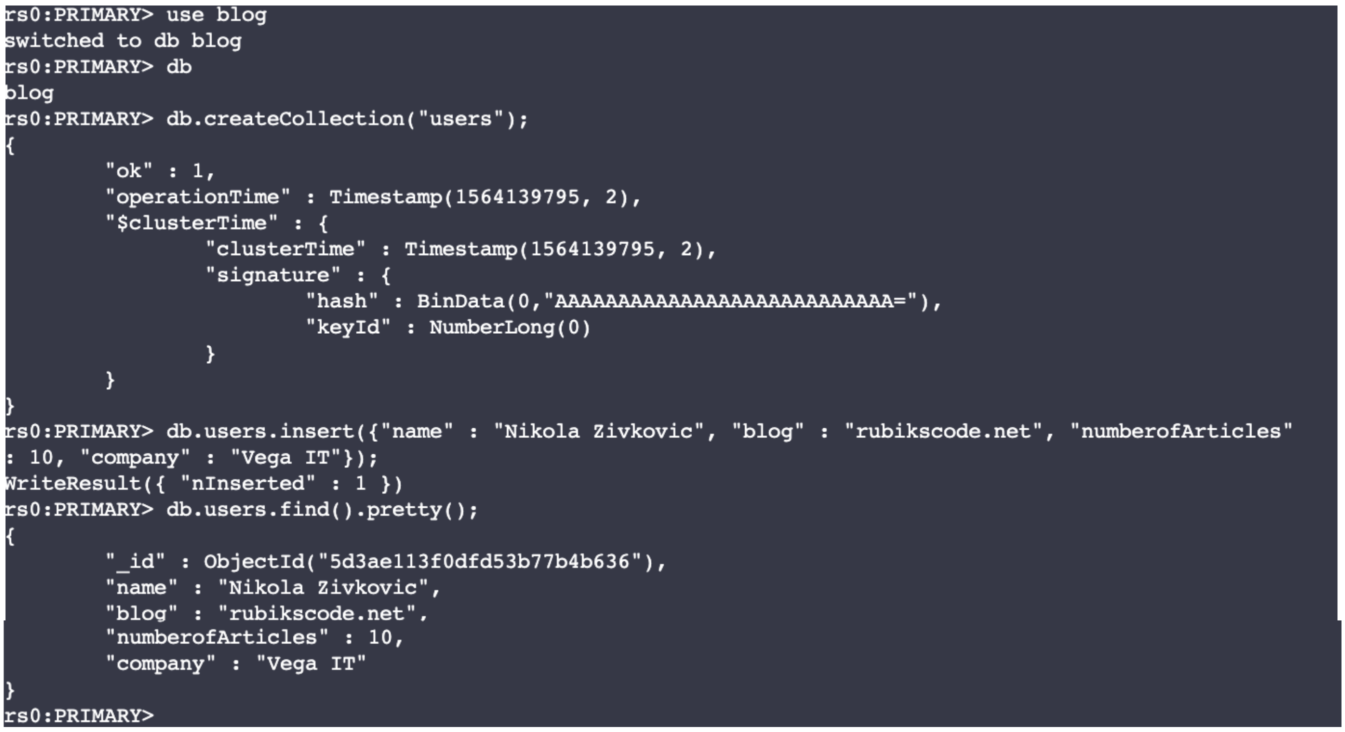
Here are the relevant commands, from that lesson, that you can type into the terminal to set up a database:

```
use blog
```

```
db
db.createCollection("users");


db.users.insert({"name" : "Nikola Zivkovic", "blog" : "rubikscode.net", "numberofArticles" :
db.users.find().pretty();
```

You should see the following result:



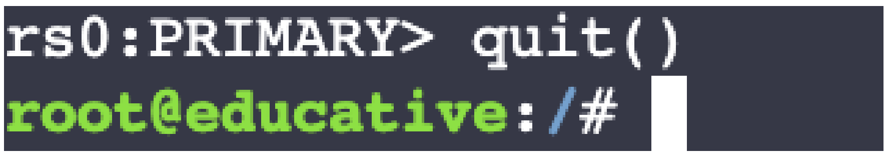
```
rs0:PRIMARY> use blog
switched to db blog
rs0:PRIMARY> db
blog
rs0:PRIMARY> db.createCollection("users");
{
        "ok" : 1,
        "operationTime" : Timestamp(1564139795, 2),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1564139795, 2),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rs0:PRIMARY> db.users.insert({"name" : "Nikola Zivkovic", "blog" : "rubikscode.net", "numberofArticles"
: 10, "company" : "Vega IT"});
WriteResult({ "nInserted" : 1 })
rs0:PRIMARY> db.users.find().pretty();
{
        "_id" : ObjectId("5d3ae113f0dfd53b77b4b636"),
        "name" : "Nikola Zivkovic",
        "blog" : "rubikscode.net",
        "numberofArticles" : 10,
        "company" : "Vega IT"
}
rs0:PRIMARY>
```

Quitting the Shell

Next, type the following command into the terminal to quit the `mongo` shell:

```
quit()
```

This command should bring you out of the shell.



```
rs0:PRIMARY> quit()
root@educative:/#
```

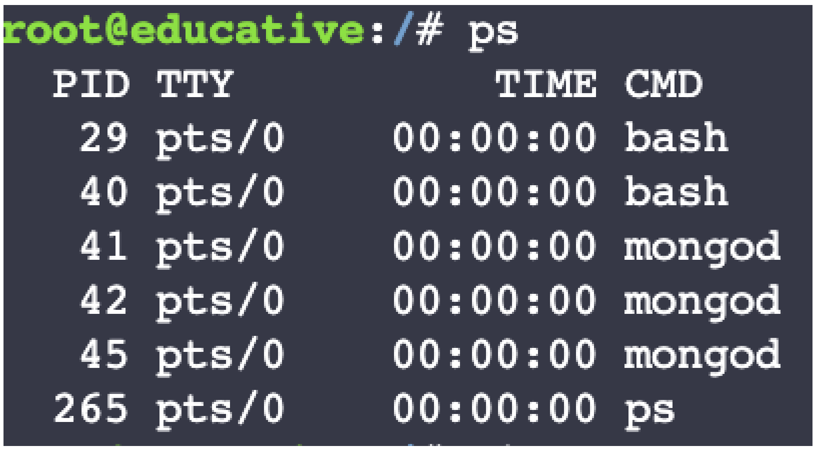
Killing a Secondary Node

Once you are out of the shell, type the following command into the terminal to see the processes that are running and their relevant PIDs:

```
ps
```

Under the `CMD` heading, you will see the three `mongod` processes running; and their respective `PID`s will be displayed under the `PID` heading.

You should see something like this:

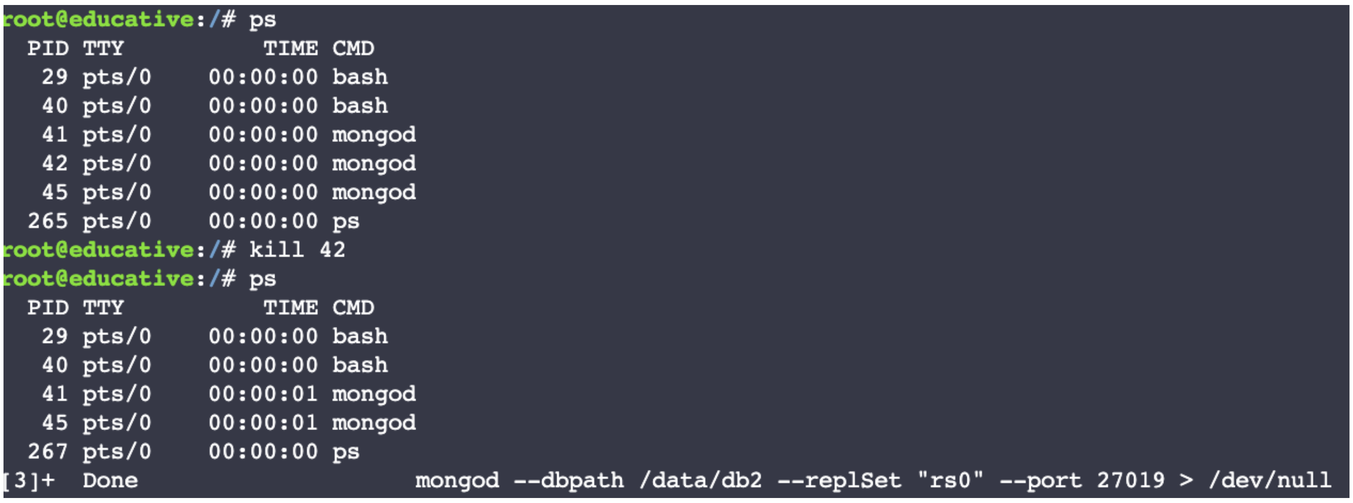


```
root@educative:/# ps
  PID TTY          TIME CMD
   29 pts/0    00:00:00 bash
   40 pts/0    00:00:00 bash
   41 pts/0    00:00:00 mongod
   42 pts/0    00:00:00 mongod
   45 pts/0    00:00:00 mongod
  265 pts/0    00:00:00 ps
```

In order to kill a *secondary* node, type the `kill PIDNumber` command, replacing `PIDNumber` with the `PID` of the *second* `mongod` entry, in the output of the `ps` command. Do this in the terminal below:

```
kill PIDNumber
```

Type the `ps` command into the terminal again; you should be able to see only *two* processes running now.



```
root@educative:/# ps
  PID TTY          TIME CMD
   29 pts/0    00:00:00 bash
   40 pts/0    00:00:00 bash
   41 pts/0    00:00:00 mongod
   42 pts/0    00:00:00 mongod
   45 pts/0    00:00:00 mongod
  265 pts/0    00:00:00 ps
root@educative:/# kill 42
root@educative:/# ps
  PID TTY          TIME CMD
   29 pts/0    00:00:00 bash
   40 pts/0    00:00:00 bash
   41 pts/0    00:00:01 mongod
   45 pts/0    00:00:01 mongod
  267 pts/0    00:00:00 ps
[3]+  Done                    mongod --dbpath /data/db2 --replSet "rs0" --port 27019 > /dev/null
```

Checking the Database

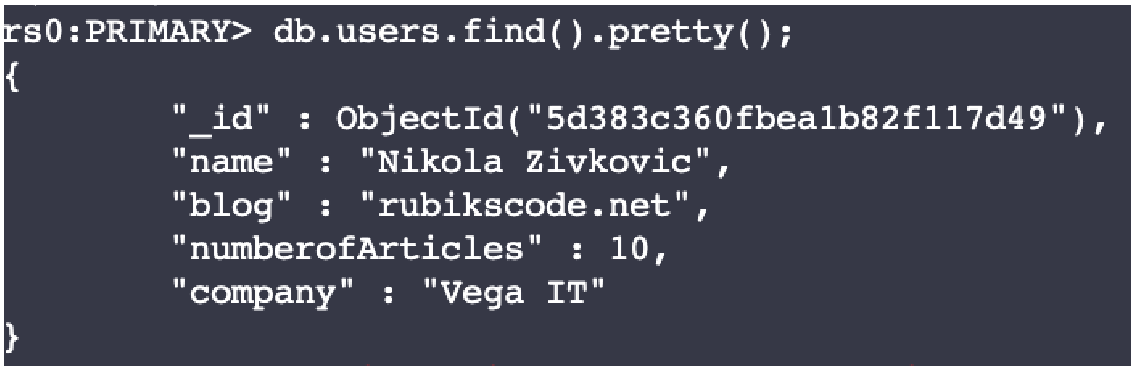
In the last step, run the command `mongo` to connect to the `mongod` process that is running on port `27017` again.

Once you are connected, type the following commands into the terminal:

```
use blog
show collections
db.users.find().pretty();
```

> **Note:** To learn more about these commands, read the following lesson.

You should see that even though one of the MongoDB servers has died, no data has been lost.



```
rs0:PRIMARY> db.users.find().pretty();
{
        "_id" : ObjectId("5d383c360fbea1b82f117d49"),
        "name" : "Nikola Zivkovic",
        "blog" : "rubikscode.net",
        "numberofArticles" : 10,
        "company" : "Vega IT"
}
```

Terminal

Try out all of the commands mentioned above in the terminal given below!

● Terminal

Go back (click here to go back above).

In the next lesson, we will look at the concept of *Sharding* in MongoDB.