04 Configuración de los miembros

Prioridad

```
clusterGetafe:PRIMARY> var config = rs.config()
clusterGetafe:PRIMARY> config
{
        "_id" : "clusterGetafe",
        "version" : 1,
        ….


clusterGetafe:PRIMARY> config.members[2].priority = 2

clusterGetafe:PRIMARY> rs.reconfig(config)
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1580481349, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1580481349, 1)
```

La configuración de prioridad entra en vigor y el primario pasará a ser el miembro 3:

```
rs.status()
{
...
        "members" : [
                {
                        "_id" : 0,
                        "name" : "localhost:27017",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 898,
                        "optime" : {
                                "ts" : Timestamp(1580481431, 1),
                                "t" : NumberLong(4)
```

```
        },
        "optimeDate" : ISODate("2020-01-31T14:37:11Z"),
        "syncingTo" : "localhost:27019",
        "syncSourceHost" : "localhost:27019",
        "syncSourceId" : 2,
        "infoMessage" : "",
        "configVersion" : 2,
        "self" : true,
        "lastHeartbeatMessage" : ""
},
{
        "_id" : 1,
        "name" : "localhost:27018",
        "health" : 1,
        "state" : 2,
        "stateStr" : "SECONDARY",
        "uptime" : 888,
        "optime" : {
                "ts" : Timestamp(1580481431, 1),
                "t" : NumberLong(4)
        },
        "optimeDurable" : {
                "ts" : Timestamp(1580481431, 1),
                "t" : NumberLong(4)
        },
        "optimeDate" : ISODate("2020-01-31T14:37:11Z"),
        "optimeDurableDate" : ISODate("2020-01-31T14:37:11Z"),
        "lastHeartbeat" : ISODate("2020-01-31T14:37:14.328Z"),
        "lastHeartbeatRecv" : ISODate("2020-01-31T14:37:15.295Z"),
        "pingMs" : NumberLong(0),
        "lastHeartbeatMessage" : "",
        "syncingTo" : "localhost:27017",
        "syncSourceHost" : "localhost:27017",
        "syncSourceId" : 0,
        "infoMessage" : "",
        "configVersion" : 2
},
{
        "_id" : 2,
        "name" : "localhost:27019",
        "health" : 1,
        "state" : 1,
        "stateStr" : "PRIMARY",
        "uptime" : 880
```

Si no conectamos a él nos devolverá el pront como PRIMARY.

Añadir un nuevo miembro a un cluster existente

En primer lugar creamos el directorio del nuevo servidor:

Linux

```
mkdir data/rs4
```

Windows

```
md c:\data\server4
```

Levantamos el servidor:

```
mongod --replSet clusterGetafe --dbpath data/server4 --port 27020 --oplogSize 200
```

Ahora desde el primario usamos el método rs.add() para añadir el servidor con:

```
clusterGetafe:PRIMARY> rs.add({
  host: "localhost:27020",
  priority: 0,
  votes: 0
})
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1580482551, 2),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1580482551, 2)
```

Comprobamos que el servidor se ha unido como secundario

```
rs.status()

                {
                        "_id" : 3,
                        "name" : "localhost:27020",
                        "health" : 1,
```

```
                        "state" : 2,
                        "stateStr" : "SECONDARY",
```

Ahora cambiamos la prioridad y votos del nuevo miembro:

```
clusterGetafe:PRIMARY> var config = rs.conf()
clusterGetafe:PRIMARY> config.members[3].priority = 1;
1
clusterGetafe:PRIMARY> config.members[3].votes = 1;
1
clusterGetafe:PRIMARY> rs.reconfig(config)
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1580482836, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1580482836, 1)
```

Comprobación de la tolerancia a fallos.

Si tenemos ahora mismo 4 servidores la tolerancia a fallos será 1. Podemos comprobar como si fallaran 2 servidores perderemos la disponibilidad de escrituras en el cluster.

Apagamos el primario actual.

Apagamos el siguiente primario elegido tras caer el anterior.

Comprobamos como los dos vivos que quedan se mantienen como secundarios y no llegan a ser primarios.

```
clusterGetafe:SECONDARY> rs.isMaster()
{
        "hosts" : [
                "localhost:27017",
                "localhost:27018",
                "localhost:27019",
                "localhost:27020"
        ],
        "setName" : "clusterGetafe",
        "setVersion" : 4,
```

```
        "ismaster" : false,
        "secondary" : true,
…
```

Si ahora levantamos los servidores volveremos a obtener primario.

Añadir un miembro árbitro

Para aumentar la tolerancia a fallos a 2 podemos añadir un miembro como árbitro

Creamos su directorio

Linux

```
mkdir data/server5
```

Windows

```
md c:\data\server5
```

Levantamos el servidor:

```
mongod --replSet clusterGetafe --dbpath data/server5 --port 27021 --oplogSize 200
```

Y desde la shell del primario lo añadimos:

```
clusterGetafe:PRIMARY> rs.addArb("localhost:27021")
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1580484412, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1580484412, 1)
```

El árbitro una vez añadido no puede ser reconfigurado para ser otro tipo de miembro

Comprobamos con:

```
rs.status()
…
```

```
                        "_id" : 4,
                        "name" : "localhost:27021",
                        "health" : 1,
                        "state" : 7,
                        "stateStr" : "ARBITER",
…
```

Si nos conectamos al árbitro con la shell podemos ver el promt:

clusterGetafe:ARBITER>

Si ahora tumbamos el primario 2 veces veremos como al tener 3 consigue un primario de entre los dos vivos y por tanto la tolerancia ha aumentado a 2:

Comprobamos:

rs.status()

En la propiedad health 1 disponible, 0 no disponible

Miembro oculto

Los miembros ocultos son aquellos que no pueden llegar a ser primarios y son invisibles para los clientes de aplicaciones. Los miembros ocultos pueden o no votar en las elecciones. Si no tuvieran votos hay que asegurarse que el cojunto de los miembros con votos constituyen mayoría.

Para configurar un miembro como oculto es necesario establecer su prioridad como 0 y la propiedad hidden como true.

En el primario:

```
clusterGetafe:PRIMARY> var config = rs.config()

clusterGetafe:PRIMARY> config.members[3].priority = 0
clusterGetafe:PRIMARY> config.members[3].hidden = true

clusterGetafe:PRIMARY> rs.reconfig(config)
```

Si ahora lanzamos isMaster() vemos como está oculto

```
clusterGetafe:PRIMARY> db.isMaster()
{
        "hosts" : [
                "localhost:27017",
                "localhost:27018",
                "localhost:27019"
        ],
        "arbiters" : [
                "localhost:27021"
        ],
```

Miembro delayed

Para configurar el miembro como delayed, debe estar como prioridad 0, hidden true y se establece la siguiente propiedad:

```
clusterGetafe:PRIMARY> config = rs.config()

clusterGetafe:PRIMARY> config.members[3].slaveDelay = 300  // En segundos
300
clusterGetafe:PRIMARY> rs.reconfig(config)
```

Ahora escribimos una operación de escritura:

```
clusterGetafe:PRIMARY> use getafeTest
switched to db getafeTest
clusterGetafe:PRIMARY> db.foo2.insert({mensaje: "Check Delay"})
```

Y la exportamos de un secundario normal y del secundario delayed:

```
MacBook-Pro-de-Pedro:~ pedro$ mongoexport --port 27017 --collection=foo2
--db=getafeTest --out=mensaje27017.json
2020-02-03T18:00:49.902+0100    connected to: mongodb://localhost:27017/
2020-02-03T18:00:49.906+0100    exported 1 record

MacBook-Pro-de-Pedro:~ pedro$ mongoexport --port 27020 --collection=foo2
--db=getafeTest --out=mensaje27020.json
2020-02-03T18:01:20.401+0100    connected to: mongodb://localhost:27020/
2020-02-03T18:01:20.401+0100    exported 0 records
```

A los 5 minutos intentamos exportar de nuevo:

```
MacBook-Pro-de-Pedro:~ pedro$ mongoexport --port 27020 --collection=foo2
--db=getafeTest --out=mensaje27020.json
2020-02-03T18:03:23.933+0100    connected to: mongodb://localhost:27020/
2020-02-03T18:03:23.936+0100    exported 1 record
```

Podemos también comprobar que en este cluster mantenemos la tolerancia 2.

Votos de los miembros

Los miembros sin votos permiten añadir miembros para distribuir los miembros de lectura de cara a mantener el límite de 7 miembros con voto.

Simplemente se trata de configurar la propiedad priority como 0 (por lo tanto no podrá llegar a ser primario) y la propiedad votes como 0.

Retirar un miembro

Para eliminar un miembro seguimos los siguientes pasos:

1.- Apagar el servidor. Por ejemplo vamos a apagar el miembro delayed y el árbitro.

2.- Desde el primario (chequear con db.isMaster()) usar el método remove(). Ahora eliminamos los dos miembros por su dominio:puerto.

```
clusterGetafe:PRIMARY> rs.remove("localhost:27020")
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1580750338, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1580750338, 1)
}
clusterGetafe:PRIMARY> rs.remove("localhost:27021")
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1580750342, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
```

```
        },
        "operationTime" : Timestamp(1580750342, 1)
}
```

Recomprobamos:

```
clusterGetafe:PRIMARY> db.isMaster()
{
        "hosts" : [
                "localhost:27017",
                "localhost:27018",
                "localhost:27019"
        ],
        "setName" : "clusterGetafe",
        "setVersion" : 9,
        "ismaster" : true,
        "secondary" : false,
…
```