Implementar Replicación en BD MongoDB

1. Implementar en MongoDB un ReplicaSet con 3 servidores que contengan la información de la BD Finanzas. Un nodo Primary, un secondary y un arbiter.

- mongod --replSet rs --dbpath /home/mauri/data/db/rs/0 --port 27017 --oplogSize 50
- mongod --replSet rs --dbpath /home/mauri/data/db/rs/1 --port 27018 --oplogSize 50
- mongod --replSet rs --dbpath /home/mauri/data/db/rs/2 --port 27019 --oplogSize 50

2. Conectarse al Nodo PRIMARY

mongo --port 27017

```
cfg = {
        _id:"rs",
        members:[
                        {_id:0, host:"localhost:27017"},
                        {_id:1, host:"localhost:27018"},
                        {_id:2, host:"localhost:27019", arbiterOnly:true}
        ]
};

rs.initiate(cfg)
```

3. Crear la db finanzas.

use finanzas

4. Ejecutar el script facts.js 4 veces para crear volumen de datos.

load("facts.js") previo cargar el archivo en la carpeta donde estoy ejecutando el primary

5. Buscar los datos insertados, en el nodo PRIMARY.

db.facturas.find()

6. Buscar los datos insertados, en el nodo SECONDARY.

db.facturas.find()

7. Realizar un ejemplo de Fault Tolerance simulando una caída del Servidor PRIMARY.

7.1. Explicar qué sucedió.

El SECONDARY pasa automáticamente a ser PRIMARY

7.2. Verificar el estado de cada servidor.

rs.status()

PRIMARY

"_id" : 0,

"name" : "localhost:27017",

"health" : 0,

"state" : 8,

"stateStr" : "(not reachable/healthy)",

"uptime" : 0,

"optime" : {

"ts" : Timestamp(0, 0),

"t" : NumberLong(-1)

},

"optimeDurable" : {

"ts" : Timestamp(0, 0),

"t" : NumberLong(-1)

},

"optimeDate" : ISODate("1970-01-01T00:00:00Z"),

"optimeDurableDate" : ISODate("1970-01-01T00:00:00Z"),

"lastHeartbeat" : ISODate("2021-09-14T18:06:18.781Z"),

"lastHeartbeatRecv" : ISODate("2021-09-14T17:56:20.195Z"),

"pingMs" : NumberLong(0),

"lastHeartbeatMessage" : "Error connecting to localhost:27017 (127.0.0.1:27017) :: caused by :: Connection refused",

"syncSourceHost" : "",

"syncSourceId" : -1,

"infoMessage" : "",

"configVersion" : 1,

"configTerm" : 1

7.3. Insertar un nuevo documento.

db.facturas.insert({X:2})

7.4. Levantar el servidor caído.

Mongo –port 27017

rs.secondaryOk() porque ahora pasa a ser secundario

7.5. Validar la información en cada servidor.

Información coincidente con nuevo primary (27018)

8. Agregar un nuevo nodo con slaveDelay de 120 segundos.

mongod --replSet rs --dbpath /home/fede/data/db/rs/3 --port 27020 --oplogSize 50

rs.add({host:"localhost:27020", priority: 0, slaveDelay: 120})

{

"ok" : 1,

"$clusterTime" : {

"clusterTime" : Timestamp(1631648726, 1),

"signature" : {

"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),

"keyId" : NumberLong(0)

}

},

"operationTime" : Timestamp(1631648726, 1)

}

rs.secondaryOk()

9. Ejecutar nuevamente el script facts.js, asegurarse antes de ejecutarlo que el nodo con slaveDelay esté actualizado igual que el PRIMARY.

9.1. Luego de ejecutado chequear el SECONDARY.

El secondary no se actualiza hasta después de los 120 segundos

9.2. Consultar el nuevo nodo y ver cuando se actualizan los datos.

Pasado los 120 segundos se refleja la información en el nuevo nodo