↘Creación de la network → redSharding

labs@hp ~ $ **docker network create redSharding**

↘Creación de 2 replicaSet → 2 shards: {rs0,rs1} de 3 nodos cada uno: rs{0,1}Mongo{0,1,2}

↘rs0

labs@hp ~ $ **docker run --net redSharding -p 4001:27017 --name rs0Mongo0 -d mongo --shardsvr --replSet rs0 --noprealloc --smallfiles --port 27017**

labs@hp ~ $ **docker run --net redSharding -p 4002:27017 --name rs0Mongo1 -d mongo --shardsvr --replSet rs0 --noprealloc --smallfiles --port 27017**

labs@hp ~ $ **docker run --net redSharding -p 4003:27017 --name rs0Mongo2 -d mongo --shardsvr --replSet rs0 --noprealloc --smallfiles --port 27017**

↘rs1

labs@hp ~ $ **docker run --net redSharding -p 4004:27017 --name rs1Mongo0 -d mongo --shardsvr --replSet rs1 --noprealloc --smallfiles --port 27017**

labs@hp ~ $ **docker run --net redSharding -p 4005:27017 --name rs1Mongo1 -d mongo --shardsvr --replSet rs1 --noprealloc --smallfiles --port 27017**

labs@hp ~ $ **docker run --net redSharding -p 4006:27017 --name rs1Mongo2 -d mongo --shardsvr --replSet rs1 --noprealloc --smallfiles --port 27017**

labs@hp ~ $ **docker stop rs{0,1}Mongo{0,1,2}**
labs@hp ~ $ **docker start rs{0,1}Mongo{0,1,2}**

↘Servidores de Configuración → cfg{0,1,2}

labs@hp ~ $ **docker run --net redSharding -p 4007:27017 --name cfg0 -d mongo --configsvr --replSet rsConfig --noprealloc --smallfiles --port 27017**

labs@hp ~ $ **docker run --net redSharding -p 4008:27017 --name cfg1 -d mongo --configsvr --replSet rsConfig --noprealloc --smallfiles --port 27017**

labs@hp ~ $ **docker run --net redSharding -p 4009:27017 --name cfg2 -d mongo --configsvr --replSet rsConfig --noprealloc --smallfiles --port 27017**

labs@hp ~ $ **docker stop cfg{0,1,2}**
labs@hp ~ $ **docker start cfg{0,1,2}**

↘Configuramos los 2 replicaSet → rs{0,1}Mong0{0,1,2}

↘rs0

labs@hp ~ $ **docker exec -ti rs0Mongo0 mongo**
> **rs.initiate()**
rs0:PRIMARY> **rs.status()**
rs0:PRIMARY> **rs.add("rs0Mongo1:27017")**
rs0:PRIMARY> **rs.add("rs0Mongo2:27017")**
rs0:PRIMARY> **rs.status()**
rs0:PRIMARY> **cfg = rs.conf()**
rs0:PRIMARY> **cfg.members[0].host = "rs0Mongo0:27017"**
rs0:PRIMARY> **rs.reconfig(cfg)**
rs0:PRIMARY> **rs.status()**

↘rs1

labs@hp ~ $ **docker exec -ti rs1Mongo0 mongo**

> **rs.initiate()**
rs1:PRIMARY> **rs.status()**
rs1:PRIMARY> **rs.add("rs1Mongo1:27017")**
rs1:PRIMARY> **rs.add("rs1Mongo2:27017")**
rs1:PRIMARY> **cfg = rs.conf()**
rs1:PRIMARY> **cfg.members[0].host = "rs1Mongo0:27017"**
rs1:PRIMARY> **rs.reconfig(cfg)**
rs1:PRIMARY> **rs.status()**

↘Configurar Servidores de Configuración → cfg{0,1,2}

labs@hp ~ $ **docker exec -ti cfg0 mongo**
> **rs.initiate()**
rsConfig:PRIMARY> **rs.status()**
rsConfig:PRIMARY> **rs.add("cfg1:27017")**
rsConfig:PRIMARY> **rs.add("cfg2:27017")**
rsConfig:PRIMARY> **cfg = rs.conf()**
rsConfig:PRIMARY> **cfg.members[0].host="cfg0:27017"**
rsConfig:PRIMARY> **rs.reconfig(cfg)**
rsConfig:PRIMARY> **rs.status()**

↘Direcciones IP de los Servidores net → redSharding :

labs@hp ~ $ **docker inspect -f "{{.NetworkSettings.Networks.redSharding.IPAddress}}""{{.Name}}" rs{0,1}Mongo{0,1,2}**

172.20.0.2/rs0Mongo0
172.20.0.3/rs0Mongo1
172.20.0.4/rs0Mongo2
172.20.0.5/rs1Mongo0
172.20.0.6/rs1Mongo1

172.20.0.7/rs1Mongo2

labs@hp ~ $ **docker inspect -f "{{.NetworkSettings.Networks.redSharding.IPAddress}}""{{.Name}}" cfg{0,1,2}**

172.20.0.8/cfg0
172.20.0.9/cfg1
172.20.0.10/cfg2

↘Máquina mongos.

labs@hp ~ $ **docker run --name mongos --net redSharding -ti mongo bash**
labs@hp ~ $ **docker stop mongos**

labs@hp ~ $ **docker start mongos**
labs@hp ~ $ **docker exec -ti mongos bash**
root@db14ad100156:/# **mongos --port 27017 --configdb rsConfig/cfg0:27017,cfg1:27017,cfg2:27017**

↘ Añadimos los Shardings → rs{0,1}/Mongo0

labs@hp ~ $ **docker exec -ti mongos mongo**
mongos> **sh.addShard("rs0/rs0Mongo0:27017")**
mongos> **sh.addShard("rs1/rs1Mongo0:27017")**
mongos> **sh.status()**
--- Sharding Status ---
 sharding version: {
 "_id" : 1,
 "minCompatibleVersion" : 5,
 "currentVersion" : 6,
 "clusterId" : ObjectId("5c6b001478757d20a468c711")
 }
 shards:
 { "_id" : "rs0", "host" : "rs0/rs0Mongo0:27017,rs0Mongo1:27017,rs0Mongo2:27017", "state" : 1 }
 { "_id" : "rs1", "host" : "rs1/rs1Mongo0:27017,rs1Mongo1:27017,rs1Mongo2:27017", "state" : 1 }
 active mongoses:
 "4.0.5" : 1
 autosplit:
 Currently enabled: yes
 balancer:
 Currently enabled: yes
 Currently running: no
 Failed balancer rounds in last 5 attempts: 0
 Migration Results for the last 24 hours:
 No recent migrations

databases:
{ "_id" : "config", "primary" : "config", "partitioned" : true }
config.system.sessions
shard key: { "_id" : 1 }
unique: false
balancing: true
chunks:
rs0 1
{ "_id" : { "$minKey" : 1 } } -->> { "_id" : { "$maxKey" : 1 } } on : rs0 Timestamp(1, 0)

↘ Método-1: Shard Keys

```
mongos> use shardedDB
mongos> sh.enableSharding("shardedDB")
mongos> db.test_collection.createIndex({_id: "hashed"})
mongos> sh.shardCollection("shardedDB.test_sharded", { "_id": "hashed" })
mongos> sh.status()
…
shardedDB.test_sharded
        shard key: { "_id" : "hashed" }
        unique: false
        balancing: true
        chunks:
            rs0     2
            rs1     2
        { "_id" : { "$minKey" : 1 } } -->> { "_id" : NumberLong("-4611686018427387902") } on : rs0 Timestamp(1, 0)
        { "_id" : NumberLong("-4611686018427387902") } -->> { "_id" : NumberLong(0) } on : rs0 Timestamp(1, 1)
        { "_id" : NumberLong(0) } -->> { "_id" : NumberLong("4611686018427387902") } on : rs1 Timestamp(1, 2)
        { "_id" : NumberLong("4611686018427387902") } -->> { "_id" : { "$maxKey" : 1 } } on : rs1 Timestamp(1, 3)
```

↘ Inserción de Datos:

```
mongos> for (var i = 1; i <= 10000; i++) db.test_sharded.insert( { x : i } )
mongos> db.test_sharded.find().count()
10000

mongos> db.test_sharded.find().limit(8)
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08427"), "x" : 3 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08428"), "x" : 4 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08429"), "x" : 5 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc0842a"), "x" : 6 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc0842c"), "x" : 8 }
```

{ "_id" : ObjectId("5c6fa14db05b24ea6dc0842d"), "x" : 9 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08430"), "x" : 12 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08432"), "x" : 14 }

↘ Comprobamos el Balanceo de Datos → **Shared Keys**:

↘rs0
rs0:PRIMARY> **use shardedDB**
rs0:PRIMARY> **db.test_sharded.count()**
5035
rs0:PRIMARY> d**b.test_sharded.find().limit(8)**
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08427"), "x" : 3 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08428"), "x" : 4 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08429"), "x" : 5 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc0842a"), "x" : 6 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc0842c"), "x" : 8 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc0842d"), "x" : 9 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08430"), "x" : 12 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08432"), "x" : 14 }

↘rs1
rs1:PRIMARY> **use shardedDB**
rs1:PRIMARY> **db.test_sharded.count()**
4965
rs1:PRIMARY> **db.test_sharded.find().limit(8)**
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08425"), "x" : 1 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08426"), "x" : 2 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc0842b"), "x" : 7 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc0842e"), "x" : 10 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc0842f"), "x" : 11 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08431"), "x" : 13 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08433"), "x" : 15 }
{ "_id" : ObjectId("5c6fa14db05b24ea6dc08434"), "x" : 16 }

↘ Método-2: Range Keys

mongos> **use rangedDB**
mongos> **sh.enableSharding("rangedDB")**
mongos> **db.test_ranged.createIndex({number:1})**
mongos> **sh.shardCollection("rangedDB.test_ranged",{"number":1})**
mongos> **sh.addShardTag("rs0","sh0")**
mongos> **sh.addShardTag("rs1","sh1")**
mongos> **sh.addTagRange("rangedDB.test_ranged", {"number":0}, {"number":10}, "sh0")**
mongos> **sh.addTagRange("rangedDB.test_ranged", {"number":100}, {"number":1000}, "sh1")**
mongos> **sh.status()**

```
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("5c6f8a40bd3c6354e86954c4")
  }
  shards:
      {  "_id" : "rs0",  "host" : "rs0/rs0Mongo0:27017,rs0Mongo1:27017,rs0Mongo2:27017",
"state" : 1,  "tags" : [ "sh0" ] }
      {  "_id" : "rs1",  "host" : "rs1/rs1Mongo0:27017,rs1Mongo1:27017,rs1Mongo2:27017",
"state" : 1,  "tags" : [ "sh1" ] }
  active mongoses:
      "4.0.6" : 1
  autosplit:
      Currently enabled: yes
  balancer:
      Currently enabled:  yes
      Currently running:  no
      Failed balancer rounds in last 5 attempts:  0
      Migration Results for the last 24 hours:
          1 : Success
  databases:
      {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
          config.system.sessions
              shard key: { "_id" : 1 }
              unique: false
              balancing: true
              chunks:
                  rs0         1
              { "_id" : { "$minKey" : 1 } } -->> { "_id" : { "$maxKey" : 1 } } on : rs0
Timestamp(1, 0)
      {  "_id" : "rangedDB",  "primary" : "rs1",  "partitioned" : true,  "version" : {  "uuid" :
UUID("52a9e970-aaf4-4c68-8024-8d5857c6e501"),  "lastMod" : 1 } }
          rangedDB.test_ranged
              shard key: { "number" : 1 }
              unique: false
              balancing: true
              chunks:
                  rs0         1
                  rs1         4
              { "number" : { "$minKey" : 1 } } -->> { "number" : 0 } on : rs1 Timestamp(2, 1)
              { "number" : 0 } -->> { "number" : 10 } on : rs0 Timestamp(2, 0)
              { "number" : 10 } -->> { "number" : 100 } on : rs1 Timestamp(1, 3)
              { "number" : 100 } -->> { "number" : 1000 } on : rs1 Timestamp(1, 4)
              { "number" : 1000 } -->> { "number" : { "$maxKey" : 1 } } on : rs1 Timestamp(1,
              5)
```

tag: sh0 { "number" : 0 } -->> { "number" : 10 }
tag: sh1 { "number" : 100 } -->> { "number" : 1000 }

↘ Inserción de Datos:

mongos> **db.test_ranged.insert({number: 8})**
mongos> **db.test_ranged.insert({number: 4})**
mongos> **db.test_ranged.insert({number: 156})**
mongos> **db.test_ranged.insert({number: 77})**
mongos> **db.test_ranged.insert({number: 770})**

↘ Comprobamos el Balanceo de Datos → **Range Keys:**

↘rs0
rs0:PRIMARY> **use rangedDB**
rs0:PRIMARY> **db.test_ranged.find()**
{ "_id" : ObjectId("5c6f948bb05b24ea6dc0841b"), "number" : 8 }
{ "_id" : ObjectId("5c6f967bb05b24ea6dc0841c"), "number" : 4 }

↘rs1
rs1:PRIMARY> **use rangedDB**
rs1:PRIMARY> **db.test_ranged.find()**
{ "_id" : ObjectId("5c6f96a5b05b24ea6dc0841d"), "number" : 156 }
{ "_id" : ObjectId("5c6f9716b05b24ea6dc0841e"), "number" : 77 }
{ "_id" : ObjectId("5c6f974fb05b24ea6dc0841f"), "number" : 770

REFERENCIAS:

https://docs.mongodb.com/manual/sharding/
https://charlascylon.com/2014-01-30-tutorial-mongodb-explicando-el-sharding-con-una
https://www.adictosaltrabajo.com/2016/11/29/sharding-en-mongodb/
https://warzycha.pl/mongo-db-sharding-docker-example/
https://www.sohamkamani.com/blog/2016/06/30/docker-mongo-replica-set/
https://openwebinars.net/cursos/mongodb/