Para probar la configuración y puesta en marcha de un cluster de bases de datos mongo que balanceen la carga y la repliquen usaremos docker. Esto simplificará el proceso y nos permitirá realizarlo todo desde la misma máquina, por ello será necesario tener instalado docker (https://docs.docker.com/engine/installation/).

El proceso, si se prefiere, puede realizarse con máquina distintas pero que sean visibles en red.

Crearemos en total 10 contenedores mongo y una subred: 2 replica set de 3 nodos, 3 servidores de configuración y un router mongos.

```
~$ docker network create openWebinarsSharding
pablo@pablo:~$ docker run --net openWebinarsSharding -p 4001:27017 --name rs0mongo0 -d mongo --
shardsvr --replSet rs0 --noprealloc --smallfiles --port 27017
2ef01a49dc74d36d68696d55ed081843133eb26448a9d35bf6a15403afadec03
pablo@pablo:~$ docker run --net openWebinarsSharding -p 4002:27017 --name rs0mongo1 -d mongo --
shardsvr --replSet rs0 --noprealloc --smallfiles --port 27017
783d9b2443f6ad5c34a2639d768e3a05bfe286fda5307019e29c28fe16d69925
pablo@pablo:~$ docker run --net openWebinarsSharding -p 4003:27017 --name rs0mongo2 -d mongo --
shardsvr --replSet rs0 --noprealloc --smallfiles --port 27017
3bf27db0e5a5a066113df63e3f0d86589f100130ef451675ff165f657c59c85a
pablo@pablo:~$ docker run --net openWebinarsSharding -p 4004:27017 --name rs1mongo0 -d mongo --
shardsvr --replSet rs1 --noprealloc --smallfiles --port 27017
fd382df68a38fc2088abd0999185d417d49577d8dca2dc787b2eecf8cd02fb57
pablo@pablo:~$ docker run --net openWebinarsSharding -p 4005:27017 --name rs1mongo1 -d mongo --
shardsvr --replSet rs1 --noprealloc --smallfiles --port 27017
001e07cf5303740a6292653375121549ead6371853c9255a0c3c60ac4ca0e307
pablo@pablo:~$ docker run --net openWebinarsSharding -p 4006:27017 --name rs1mongo2 -d mongo --
shardsvr --replSet rs1 --noprealloc --smallfiles --port 27017
fa46725b61cfa34b8b3a669b6d029277588e46470cc2aa767618b90814946223
pablo@pablo:~$ docker run --net openWebinarsSharding -p 4007:27017 --name cfg0 -d mongo --confi
gsvr --replSet rsConfig --noprealloc --smallfiles --port 27017
8f28b55ef2adb8653489343ae323a096f123227e8a3a98a855cbe80e25664d06
pablo@pablo:~$ docker run --net openWebinarsSharding -p 4008:27017 --name cfg1 -d mongo --confi
gsvr --replSet rsConfig --noprealloc --smallfiles --port 27017
6952cf7fa6098539962697e350267a3ad59444553fea3a79075e66b8cedf0c0a
pablo@pablo:~$ docker run --net openWebinarsSharding -p 4009:27017 --name cfg2 -d mongo --confi

gsvr --replSet rsConfig --noprealloc --smallfiles --port 27017
114f57368622df75f3d8074e03e01b1ee15d1aab2eb2a86f83019b95df615b50
```

Configuramos el replSet rs0

```
~$ docker exec -ti rs0mongo0 mongo
> rs.initiate()
rs.add("rs0mongo1:27017")
rs.add("rs0mongo2:27017")
rs.status()
cfg = rs.conf()
```

```
cfg.members[0].host="rs0mongo0:27017"
rs.reconfig(cfg)
rs.status()
```

## Configuramos el replSet rs01

```
~$ docker exec -ti rs1mongo0 mongo
> rs.initiate()
rs.add("rs1mongo1:27017")
rs.add("rs1mongo2:27017")
rs.status()
cfg = rs.conf()
cfg.members[0].host="rs1mongo0:27017"
rs.reconfig(cfg)
rs.status()
```

## Configuramos el replSet rsConfig

```
~$ docker exec -ti rs1mongo0 mongo
> rs.initiate()
rs.add("cfg1:27017")
rs.add("cfg2:27017")
rs.status()
cfg = rs.conf()
cfg.members[0].host="cfg0:27017"
rs.reconfig(cfg)
rs.status()
```

## Levantamos la máquina mongos

```
docker run --name mongos --net openWebinarsSharding -ti mongo bash
```

## En la propia máquina, levantamos un mongos:

```
mongos --port 27017 --configdb <IP_of_container_cfg1>:27017, <IP_of_container_cfg2>:27017, <IP_of_container_cfg3>:27017
Accediendo al contenedor (docker exec -ti mongos mongo), ejecutamos el cliente mongo:
sh.addShard("REPLICASETNAME/<IP_of_rs1_srv1>:27017")
sh.addShard("REPLICASETNAME/<IP_of_rs2_srv1>:27017")
sh.status()
Hashed key
Use shardedDB
db.test_collection.ensureIndex( { _id : "hashed" } )
sh.enableSharding("shardedDB")
sh.shardCollection("test_db.test_collection", { "_id": "hashed" } )
for (var i = 1; i <= 500; i++) db.test_collection.insert( { x : i } )
Comprobar donde se han guardado (en los replSet). Es visible que el sharding se realiza correctamente.
Range key
Use ordenada
sh.enableSharding("ordenada")
```

```
db.test.ensureIndex({number:1})
sh.shardCollection("ordenada.test",{"number":1})
sh.status()
sh.addShardTag("rs0","sh0")
sh.addShardTag("rs1","sh1")
sh.addTagRange("ordenada.test", {"number":0}, {"number":10}, "sh0")
sh.addTagRange("ordenada.test", {"number":100}, {"number":1000}, "sh0")
db.test.insert({number:8})
db.test.insert({number:80})
db.test.insert({number:48})
db.test.insert({number:800})
```

Comprobar donde se han guardado. Es visible que el sharding se realiza correctamente.