Convert a Replica Set to a Replicated Sharded Cluster

Overview

Following this tutorial, you will convert a single 3-member replica set to a cluster that consists of 2 shards. Each shard will consist of an independent 3-member replica set.

The tutorial uses a test environment running on a local system UNIX-like system. You should feel encouraged to “follow along at home.” If you need to perform this process in a production environment, notes throughout the document indicate procedural differences.

The procedure, from a high level, is as follows:

1. Create or select a 3-member replica set and insert some data into a collection.
2. Start the config databases and create a cluster with a single shard.
3. Create a second replica set with three new [mongod](http://www.dba86.com/docs/mongo/2.4/reference/program/mongod.html#bin.mongod) instances.
4. Add the second replica set as a shard in the cluster.
5. Enable sharding on the desired collection or collections.

Process

Install MongoDB according to the instructions in the [MongoDB Installation Tutorial](http://www.dba86.com/docs/mongo/2.4/installation.html#tutorials-installation).

**Deploy a Replica Set with Test Data**

If have an existing MongoDB [replica set](http://www.dba86.com/docs/mongo/2.4/reference/glossary.html#term-replica-set) deployment, you can omit the this step and continue from[Deploy Sharding Infrastructure](http://www.dba86.com/docs/mongo/2.4/tutorial/convert-replica-set-to-replicated-shard-cluster.html#convert-replica-set-to-shard-cluster-deploy-sharding-infrastructure).

Use the following sequence of steps to configure and deploy a replica set and to insert test data.

1. Create the following directories for the first replica set instance, named firstset:
   * /data/example/firstset1
   * /data/example/firstset2
   * /data/example/firstset3

To create directories, issue the following command:

mkdir -p /data/example/firstset1 /data/example/firstset2 /data/example/firstset3

1. In a separate terminal window or GNU Screen window, start three [mongod](http://www.dba86.com/docs/mongo/2.4/reference/program/mongod.html#bin.mongod) instances by running each of the following commands:
2. mongod --dbpath /data/example/firstset1 --port 10001 --replSet firstset --oplogSize 700 --rest
3. mongod --dbpath /data/example/firstset2 --port 10002 --replSet firstset --oplogSize 700 --rest
4. mongod --dbpath /data/example/firstset3 --port 10003 --replSet firstset --oplogSize 700 --rest

**NOTE**

The *--oplogSize 700* option restricts the size of the operation log (i.e. oplog) for each [mongod](http://www.dba86.com/docs/mongo/2.4/reference/program/mongod.html#bin.mongod)instance to 700MB. Without the *--oplogSize* option, each [mongod](http://www.dba86.com/docs/mongo/2.4/reference/program/mongod.html#bin.mongod) reserves approximately 5% of the free disk space on the volume. By limiting the size of the oplog, each instance starts more quickly. Omit this setting in production environments.

1. In a [mongo](http://www.dba86.com/docs/mongo/2.4/reference/program/mongo.html#bin.mongo) shell session in a new terminal, connect to the mongodb instance on port 10001 by running the following command. If you are in a production environment, first read the note below.
2. mongo localhost:10001/admin

**NOTE**

Above and hereafter, if you are running in a production environment or are testing this process with [mongod](http://www.dba86.com/docs/mongo/2.4/reference/program/mongod.html#bin.mongod) instances on multiple systems, replace “localhost” with a resolvable domain, hostname, or the IP address of your system.

1. In the [mongo](http://www.dba86.com/docs/mongo/2.4/reference/program/mongo.html#bin.mongo) shell, initialize the first replica set by issuing the following command:
2. db.runCommand({"replSetInitiate" :
3. {"\_id" : "firstset", "members" : [{"\_id" : 1, "host" : "localhost:10001"},
4. {"\_id" : 2, "host" : "localhost:10002"},
5. {"\_id" : 3, "host" : "localhost:10003"}
6. ]}})
7. {
8. "info" : "Config now saved locally. Should come online in about a minute.",
9. "ok" : 1
10. }
11. In the [mongo](http://www.dba86.com/docs/mongo/2.4/reference/program/mongo.html#bin.mongo) shell, create and populate a new collection by issuing the following sequence of JavaScript operations:
12. use test
13. switched to db test
14. people = ["Marc", "Bill", "George", "Eliot", "Matt", "Trey", "Tracy", "Greg", "Steve", "Kristina", "Katie", "Jeff"];
15. **for**(**var** i=0; i<1000000; i++){
16. name = people[Math.floor(Math.random()\*people.length)];
17. user\_id = i;
18. **boolean** = [**true**, **false**][Math.floor(Math.random()\*2)];
19. added\_at = **new** Date();
20. number = Math.floor(Math.random()\*10001);
21. db.test\_collection.save({"name":name, "user\_id":user\_id, "boolean": **boolean**, "added\_at":added\_at, "number":number });
22. }

The above operations add one million documents to the collection test\_collection. This can take several minutes, depending on your system.

The script adds the documents in the following form:

{ "\_id" : ObjectId("4ed5420b8fc1dd1df5886f70"), "name" : "Greg", "user\_id" : 4, "boolean" : **true**, "added\_at" : ISODate("2011-11-29T20:35:23.121Z"), "number" : 74 }

**Deploy Sharding Infrastructure**

This procedure creates the three config databases that store the cluster’s metadata.

**NOTE**

For development and testing environments, a single config database is sufficient. In production environments, use three config databases. Because config instances store only the *metadata* for the sharded cluster, they have minimal resource requirements.

1. Create the following data directories for three [config database](http://www.dba86.com/docs/mongo/2.4/reference/glossary.html#term-config-database) instances:
   * /data/example/config1
   * /data/example/config2
   * /data/example/config3

Issue the following command at the system prompt:

mkdir -p /data/example/config1 /data/example/config2 /data/example/config3

1. In a separate terminal window or GNU Screen window, start the config databases by running the following commands:
2. mongod --configsvr --dbpath /data/example/config1 --port 20001
3. mongod --configsvr --dbpath /data/example/config2 --port 20002
4. mongod --configsvr --dbpath /data/example/config3 --port 20003
5. In a separate terminal window or GNU Screen window, start [mongos](http://www.dba86.com/docs/mongo/2.4/reference/program/mongos.html#bin.mongos) instance by running the following command:
6. mongos --configdb localhost:20001,localhost:20002,localhost:20003 --port 27017 --chunkSize 1

**NOTE**

If you are using the collection created earlier or are just experimenting with sharding, you can use a small *--chunkSize* (1MB works well.) The default [chunkSize](http://www.dba86.com/docs/mongo/2.4/reference/configuration-options.html#chunkSize) of 64MB means that your cluster must have 64MB of data before the MongoDB’s automatic sharding begins working.

In production environments, do not use a small shard size.

The [configdb](http://www.dba86.com/docs/mongo/2.4/reference/configuration-options.html#configdb) options specify the *configuration databases* (e.g. localhost:20001, localhost:20002, and localhost:2003). The [mongos](http://www.dba86.com/docs/mongo/2.4/reference/program/mongos.html#bin.mongos) instance runs on the default “MongoDB” port (i.e. 27017), while the databases themselves are running on ports in the 30001series. In the this example, you may omit the *--port 27017* option, as 27017 is the default port.

1. Add the first shard in [mongos](http://www.dba86.com/docs/mongo/2.4/reference/program/mongos.html#bin.mongos). In a new terminal window or GNU Screen session, add the first shard, according to the following procedure:
   * Connect to the [mongos](http://www.dba86.com/docs/mongo/2.4/reference/program/mongos.html#bin.mongos) with the following command:
   * mongo localhost:27017/admin
   * Add the first shard to the cluster by issuing the [addShard](http://www.dba86.com/docs/mongo/2.4/reference/command/addShard.html#dbcmd.addShard) command:
   * db.runCommand( { addShard : "firstset/localhost:10001,localhost:10002,localhost:10003" } )
   * Observe the following message, which denotes success:
   * { "shardAdded" : "firstset", "ok" : 1 }

**Deploy a Second Replica Set**

This procedure deploys a second replica set. This closely mirrors the process used to establish the first replica set above, omitting the test data.

1. Create the following data directories for the members of the second replica set, named secondset:
   * /data/example/secondset1
   * /data/example/secondset2
   * /data/example/secondset3
2. In three new terminal windows, start three instances of [mongod](http://www.dba86.com/docs/mongo/2.4/reference/program/mongod.html#bin.mongod) with the following commands:
3. mongod --dbpath /data/example/secondset1 --port 10004 --replSet secondset --oplogSize 700 --rest
4. mongod --dbpath /data/example/secondset2 --port 10005 --replSet secondset --oplogSize 700 --rest
5. mongod --dbpath /data/example/secondset3 --port 10006 --replSet secondset --oplogSize 700 --rest

**NOTE**

As above, the second replica set uses the smaller [oplogSize](http://www.dba86.com/docs/mongo/2.4/reference/configuration-options.html#oplogSize) configuration. Omit this setting in production environments.

1. In the [mongo](http://www.dba86.com/docs/mongo/2.4/reference/program/mongo.html#bin.mongo) shell, connect to one mongodb instance by issuing the following command:
2. mongo localhost:10004/admin
3. In the [mongo](http://www.dba86.com/docs/mongo/2.4/reference/program/mongo.html#bin.mongo) shell, initialize the second replica set by issuing the following command:
4. db.runCommand({"replSetInitiate" :
5. {"\_id" : "secondset",
6. "members" : [{"\_id" : 1, "host" : "localhost:10004"},
7. {"\_id" : 2, "host" : "localhost:10005"},
8. {"\_id" : 3, "host" : "localhost:10006"}
9. ]}})
10. {
11. "info" : "Config now saved locally. Should come online in about a minute.",
12. "ok" : 1
13. }
14. Add the second replica set to the cluster. Connect to the [mongos](http://www.dba86.com/docs/mongo/2.4/reference/program/mongos.html#bin.mongos) instance created in the previous procedure and issue the following sequence of commands:
15. use admin
16. db.runCommand( { addShard : "secondset/localhost:10004,localhost:10005,localhost:10006" } )

This command returns the following success message:

{ "shardAdded" : "secondset", "ok" : 1 }

1. Verify that both shards are properly configured by running the [listShards](http://www.dba86.com/docs/mongo/2.4/reference/command/listShards.html#dbcmd.listShards) command. View this and example output below:
2. db.runCommand({listShards:1})
3. {
4. "shards" : [
5. {
6. "\_id" : "firstset",
7. "host" : "firstset/localhost:10001,localhost:10003,localhost:10002"
8. },
9. {
10. "\_id" : "secondset",
11. "host" : "secondset/localhost:10004,localhost:10006,localhost:10005"
12. }
13. ],
14. "ok" : 1
15. }

**Enable Sharding**

MongoDB must have [sharding](http://www.dba86.com/docs/mongo/2.4/reference/glossary.html#term-sharding) enabled on *both* the database and collection levels.

**Enabling Sharding on the Database Level**

Issue the [enableSharding](http://www.dba86.com/docs/mongo/2.4/reference/command/enableSharding.html#dbcmd.enableSharding) command. The following example enables sharding on the “test” database:

db.runCommand( { enableSharding : "test" } )

{ "ok" : 1 }

**Create an Index on the Shard Key**

MongoDB uses the shard key to distribute documents between shards. Once selected, you cannot change the shard key. Good shard keys:

* have values that are evenly distributed among all documents,
* group documents that are often accessed at the same time into contiguous chunks, and
* allow for effective distribution of activity among shards.

Typically shard keys are compound, comprising of some sort of hash and some sort of other primary key. Selecting a shard key depends on your data set, application architecture, and usage pattern, and is beyond the scope of this document. For the purposes of this example, we will shard the “number” key. This typically would *not* be a good shard key for production deployments.

Create the index with the following procedure:

use test

db.test\_collection.ensureIndex({number:1})

**SEE ALSO**

The [Shard Key Overview](http://www.dba86.com/docs/mongo/2.4/core/sharding-shard-key.html#sharding-shard-key) and [Shard Key](http://www.dba86.com/docs/mongo/2.4/core/sharding-shard-key.html#sharding-internals-shard-keys) sections.

**Shard the Collection**

Issue the following command:

use admin

db.runCommand( { shardCollection : "test.test\_collection", key : {"number":1} })

{ "collectionsharded" : "test.test\_collection", "ok" : 1 }

The collection test\_collection is now sharded!

Over the next few minutes the Balancer begins to redistribute chunks of documents. You can confirm this activity by switching to the test database and running [db.stats()](http://www.dba86.com/docs/mongo/2.4/reference/method/db.stats.html#db.stats) or[db.printShardingStatus()](http://www.dba86.com/docs/mongo/2.4/reference/method/db.printShardingStatus.html#db.printShardingStatus).

As clients insert additional documents into this collection, [mongos](http://www.dba86.com/docs/mongo/2.4/reference/program/mongos.html#bin.mongos) distributes the documents evenly between the shards.

In the [mongo](http://www.dba86.com/docs/mongo/2.4/reference/program/mongo.html#bin.mongo) shell, issue the following commands to return statics against each cluster:

use test

db.stats()

db.printShardingStatus()

Example output of the [db.stats()](http://www.dba86.com/docs/mongo/2.4/reference/method/db.stats.html#db.stats) command:

{

"raw" : {

"firstset/localhost:10001,localhost:10003,localhost:10002" : {

"db" : "test",

"collections" : 3,

"objects" : 973887,

"avgObjSize" : 100.33173458522396,

"dataSize" : 97711772,

"storageSize" : 141258752,

"numExtents" : 15,

"indexes" : 2,

"indexSize" : 56978544,

"fileSize" : 1006632960,

"nsSizeMB" : 16,

"ok" : 1

},

"secondset/localhost:10004,localhost:10006,localhost:10005" : {

"db" : "test",

"collections" : 3,

"objects" : 26125,

"avgObjSize" : 100.33286124401914,

"dataSize" : 2621196,

"storageSize" : 11194368,

"numExtents" : 8,

"indexes" : 2,

"indexSize" : 2093056,

"fileSize" : 201326592,

"nsSizeMB" : 16,

"ok" : 1

}

},

"objects" : 1000012,

"avgObjSize" : 100.33176401883178,

"dataSize" : 100332968,

"storageSize" : 152453120,

"numExtents" : 23,

"indexes" : 4,

"indexSize" : 59071600,

"fileSize" : 1207959552,

"ok" : 1

}

Example output of the [db.printShardingStatus()](http://www.dba86.com/docs/mongo/2.4/reference/method/db.printShardingStatus.html#db.printShardingStatus) command:

--- Sharding Status ---

sharding version: { "\_id" : 1, "version" : 3 }

shards:

{ "\_id" : "firstset", "host" : "firstset/localhost:10001,localhost:10003,localhost:10002" }

{ "\_id" : "secondset", "host" : "secondset/localhost:10004,localhost:10006,localhost:10005" }

databases:

{ "\_id" : "admin", "partitioned" : **false**, "primary" : "config" }

{ "\_id" : "test", "partitioned" : **true**, "primary" : "firstset" }

test.test\_collection chunks:

secondset 5

firstset 186

[...]

In a few moments you can run these commands for a second time to demonstrate that [chunks](http://www.dba86.com/docs/mongo/2.4/reference/glossary.html#term-chunk) are migrating from firstset to secondset.

When this procedure is complete, you will have converted a replica set into a cluster where each shard is itself a replica set.