# **Sharding**

Sharding is a method for distributing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations. Database systems with large data sets or high throughput applications can challenge the capacity of a single server. For example, high query rates can exhaust the CPU capacity of the server. Working set sizes larger than the system's RAM stress the I/O capacity of disk drives. There are two methods for addressing system growth: vertical and horizontal scaling. (Citation must be added - <https://docs.mongodb.com/manual/sharding/>)

**Vertical Scaling**

Vertical Scaling involves increasing the capacity of a single server, such as using a more powerful CPU, adding more RAM, or increasing the amount of storage space. Limitations in available technology may restrict a single machine from being sufficiently powerful for a given workload. Additionally, Cloud-based providers have hard ceilings based on available hardware configurations. As a result, there is a practical maximum for vertical scaling.

**Horizontal Scaling**

Horizontal Scaling involves dividing the system dataset and load over multiple servers, adding additional servers to increase capacity as required. While the overall speed or capacity of a single machine may not be high, each machine handles a subset of the overall workload, potentially providing better efficiency than a single high-speed high-capacity server. Expanding the capacity of the deployment only requires adding additional servers as needed, which can be a lower overall cost than high-end hardware for a single machine. The tradeoff is increased complexity in infrastructure and maintenance for the deployment.

A MongoDB sharded cluster consists of the following components:

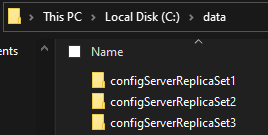
* **shard:** Each shard contains a subset of the sharded data & can be deployed as a replica set.
* **mongos:** The mongos acts as a query router, providing an interface between client applications and the sharded cluster. Starting in MongoDB 4.4, mongos can support hedged reads to minimize latencies.
* **config servers:** Config servers store metadata and configuration settings for the cluster.

**Implementation**

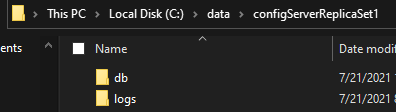
The plan is to create three shards of the database and have three replica sets for each of the shards. Then we configure a routing mechanism to route calls to any of the shards and configure the config server which store metadata for the routing to work. Also we intend to have three replica sets of the config server to ensure optimum availability of data throughout.

**Starting mongd instances Config Server**

* Creating three folders for each the replica set of the config server at location = C:\data



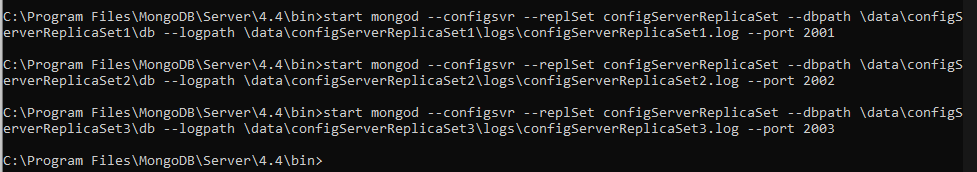
* Each of the three folders from above will contain two more folders to store the logs and db details



* Open command prompt from the bin folder of the MongoDB server (in typical installations found at - C:\Program Files\MongoDB\Server\4.4\bin)

**Note:** The version will vary depending on the installation. Ex:- 3.6, 4.2, 4.4, 5.0

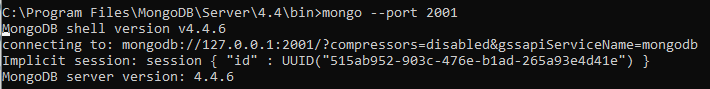
* Run the following command to start a mongodb instance to work as a configuration server for all three config server replica sets you wish to utilize



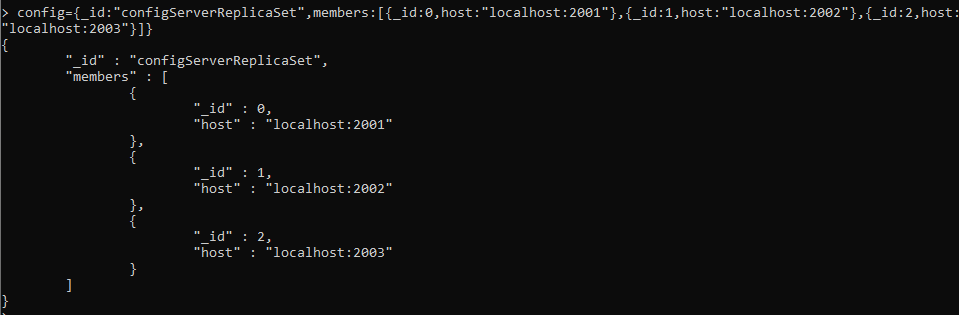
* --configsvr indicates that we wish to start a mongodb instance which behaves as a configuration server
* --replSet indicates that we wish to start this configuration server as a replica set
* “configServerReplicaSet” is the name using which we identify all three replica sets as a whole
* --dbpath followed by path indicate the location where all db related files are to be stored
* --logpath followed by path indicate the file name and location to store the logs of the server
* All three instances must run on separate ports, hence why we chose three different ports.

**Configuring the configuration server instances as replica sets**

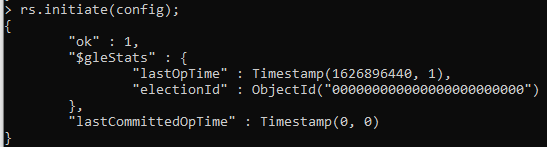
* **Note:** All these steps have to be performed on the port you wish to be the primary set
* Identify the port on which you want to run the primary set on (in our case 2001)
* Connect to the port by running the following (run from the server bin folder)



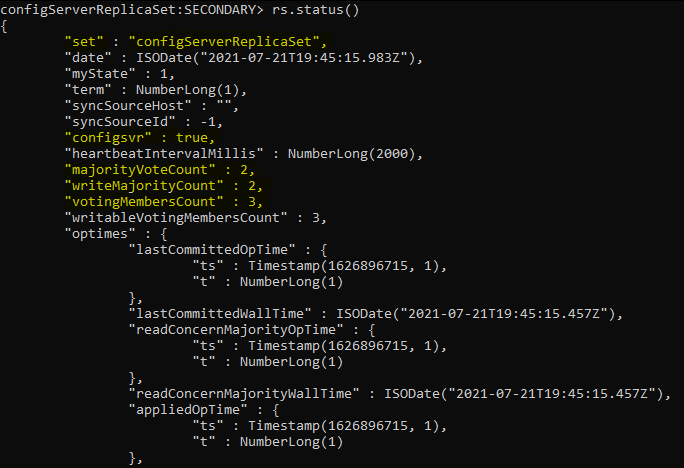
* A connection is established as seen in the above screen snippet
* The configuration parameters are entered and then mongo confirms the same by pretty printing them as seen below



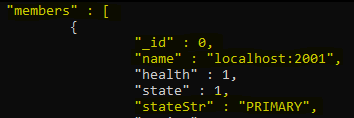
* Now we need to initiate the configuration and this can be done by running rs.initiate(config)

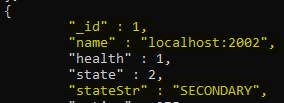


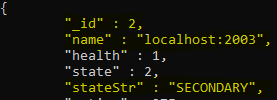
* **Note:** The terminal might show that it’s currently secondary and this might happen from time to time.
* We can check the status of the replica set by running rs.status()



* Highlighted in the screen snippet above are the set name, the confirmation that this set is being utilized as the configuration server, the count of members and majority vote count
* Scrolling the output returned will show us the members details as follows



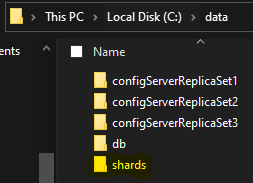




* This indicates that the set at port 2001 is acting as the primary and the sets at 2002 and 2003 are the secondary sets.
* We can validate the working of the secondary sets by connecting to them like we did for the port 2001 and by running the rs.status() command and the results should be similar to the above
* This completes the configuration of the configuration server as replica sets

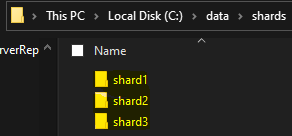
**Creating mongo instances to run as shards of the database**

* Create a folder to hold all the shards at location = C:\data

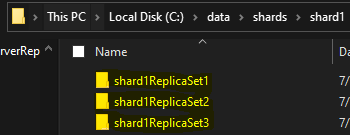


* Create three separate folders to hold the data of the shards

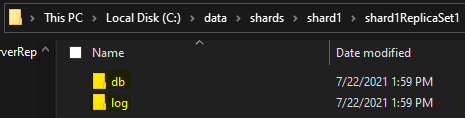
**Note:** The number of folders depends on the number of shards desired



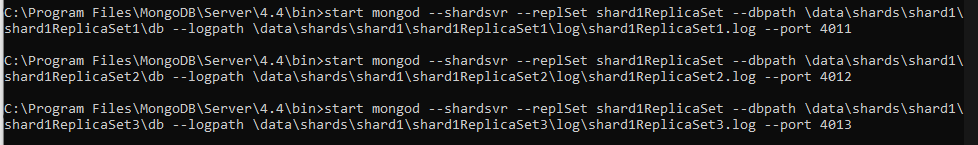
* Each of the shard folder contains three sub folders for each of its replica set



* Each shard replica folder has sub folders of db and logs as previously created for config server replica sets



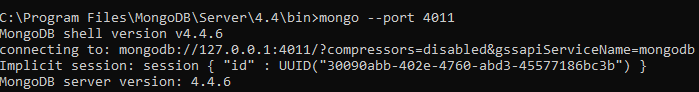
* We start the mongod instances for the shard server using the following command for each of the shard



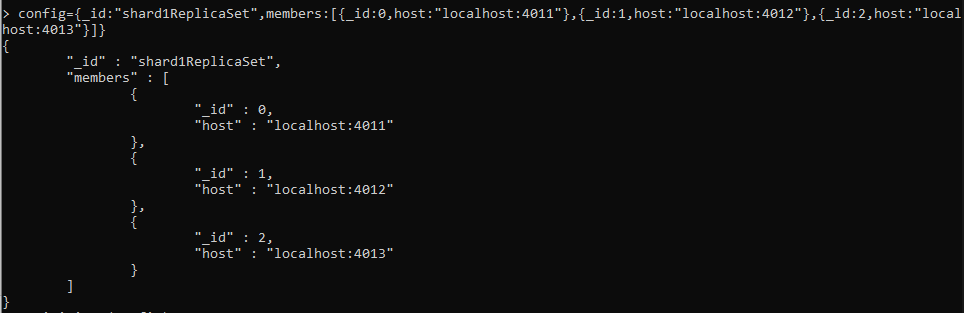
* --shardsvr indicates that we wish to start a mongd instance which behaves as a shard server
* --replSet indicates that we wish to start this configuration server as a replica set
* “shardReplicaSet” is the name using which we identify all three replica sets as a whole
* --dbpath followed by path indicate the location where all db related files are to be stored
* --logpath followed by path indicate the file name and location to store the logs of the server
* All three instances must run on separate ports, hence why we chose three different ports.

**Configuring the shard server instances as Replica Sets**

* **Note:** All these steps have to be performed on the port you wish to be the primary set
* Identify the port on which you want to run the primary set on (in our case 4011)
* Connect to the port by running the following (run from the server bin folder)



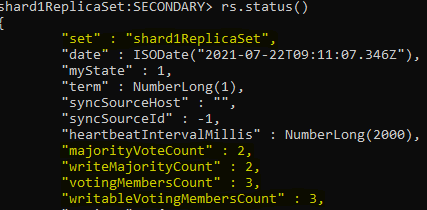
* A connection is established as seen in the above screen snippet
* The configuration parameters are entered and then mongo confirms the same by pretty printing them as seen below



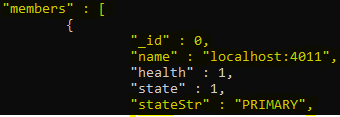
* Now we need to initiate the configuration and this can be done by running rs.initiate(config)

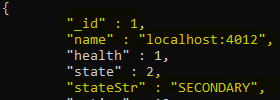


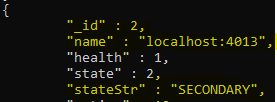
* We can check the status of the replica set by running rs.status()



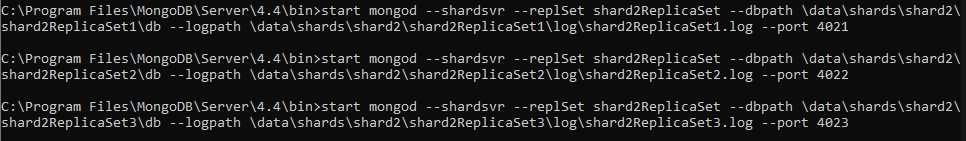
* Scrolling the output returned will show us the members details as follows

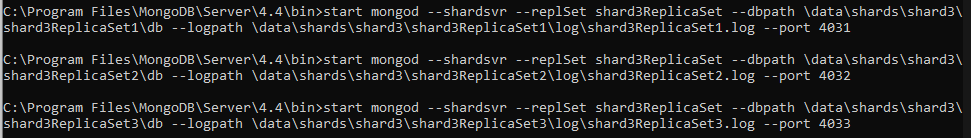






* Now we start the instances of the other shards similarly





* Configure them as we have done above

**Start a Routing Server**

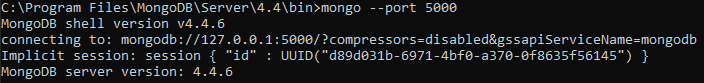
* We start a routing server - mongos as follows



* The mongos is started on the port 5000
* We indicate that “configServerReplicaSet” is the configuration file and it is at /localhost:2001 indicated by the --configdb command

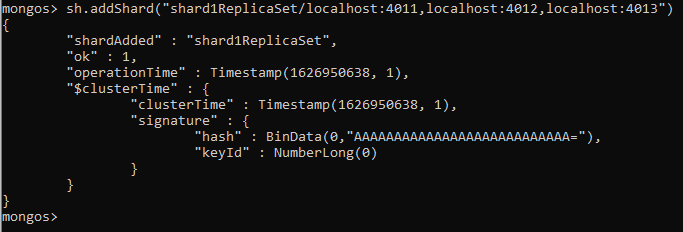
**Connect to MongoS**

* We can connect to mongo using

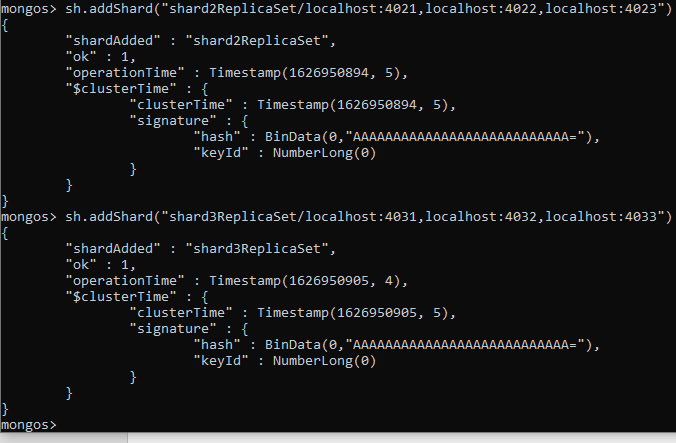


**Adding shards to the Router**

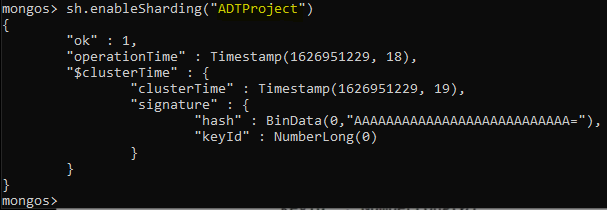
* In mongos, we can add shards using sh.addShard() command.
* Ordinarily if we did not have replica sets of the shard we could run the following command
  + sh.addShard(“localhost:4011”) [Assuming that shard server was started on 4011 port]
* Because we have the shard as replica sets we need to add shard as follows



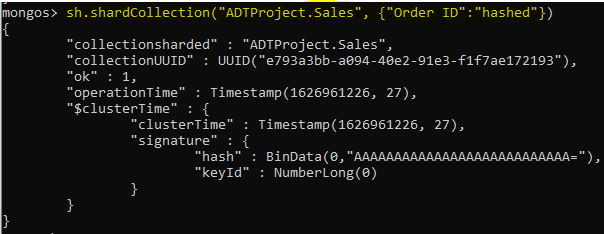
* “shard1ReplicaSet” is the name of our shard
* We append the shard name by “/” and giving all the ports of the replica sets involved
* Let’s add the remaining two shards as well



* This completes the setup of the configuration server (& it’s replica sets), shards (& it’s replica sets)
* We next have to enable the sharding on the Database of our choice and this is done as follows



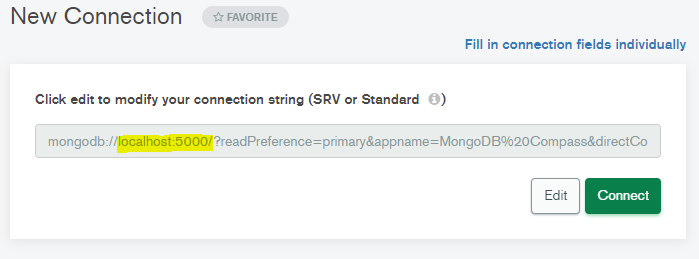
* “ADTProject” is the name of the Database we anticipate to shard
* Database consists of a collection and we need to specify the collection to be sharded as there can be a combination where a few collections of a database are sharded but the others are not sharded.
* Collection sharding can be performed by executing the following command



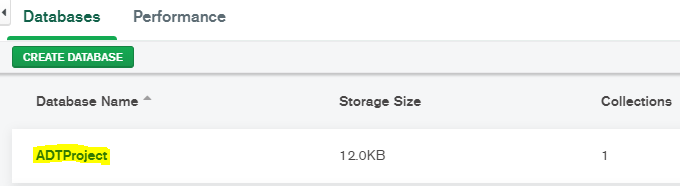
* “Sales” is the collection name we wish to shard
* We want to perform sharding on the attribute “Order ID”. Therefore, shard key is Order ID [We need info the data to execute this step]
* MongoDB facilitates two forms of sharding – Hash based and Range based
* Selection of the type of hashing is essential in improving performance [review hashing techniques and the data to be hashed at depth to finalize shard key and type]
* In case range based sharding is to be performed then the command would be as follows
  + sh.shardCollection(“ADTProject.Sales”, {“Order ID”: 1})
  + This indicates that shard key is Order ID and having 1 after colon specifies range based sharding
* The commands executed above have created the database and the collection we specified
* For importing the data, we will use the data set (.csv file we have)

**Importing data into the Database created**

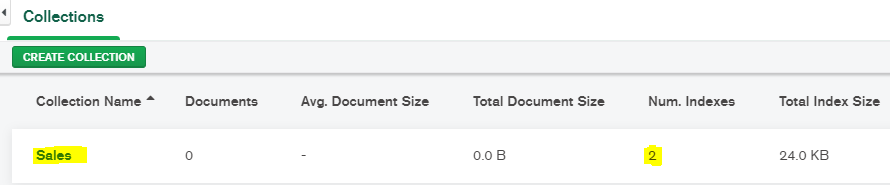
* Open MongoDB Compass
* Connect to the port 5000 directly as our routing server was configured on the same (above steps)



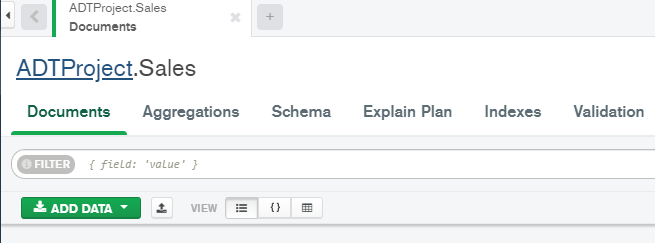
* We should be able to see the ADTProject database created post successfully connecting



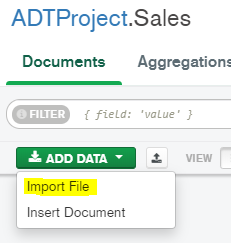
* We can click on the database name to verify the collections



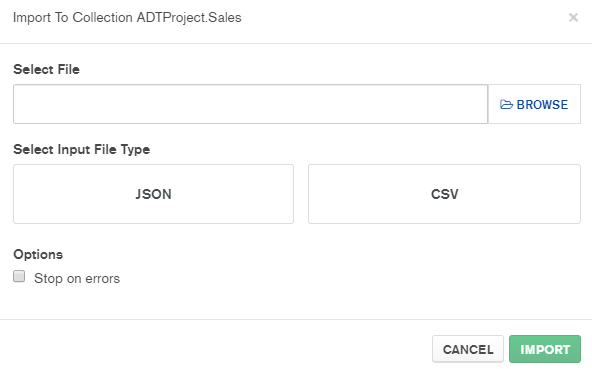
* Click on the collection name to get to the screen shown below



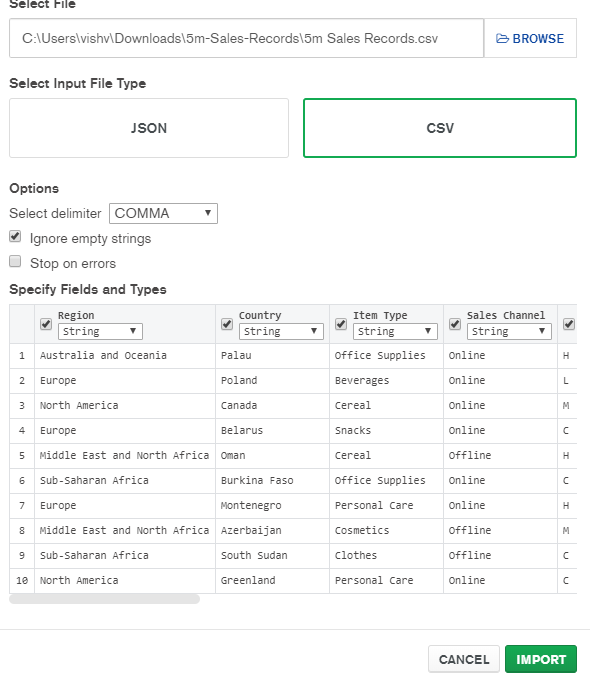
* Click on the add data button from the above screen



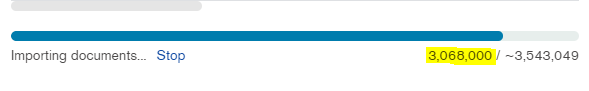
* Click on the import File option



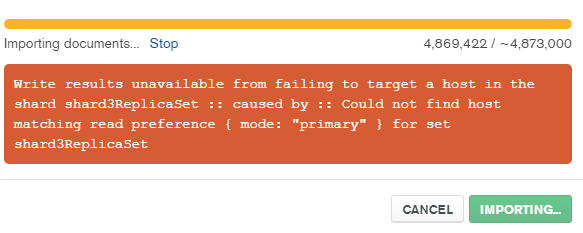
* Browse to the location of the data set, select the input type and click on import
* A preview of the data is displayed and MongoDB Compass asks you to confirm the import
* **Note:** The data set we are sharding here contains 5 million rows of Sales data and it takes a considerable amount of time to import the same



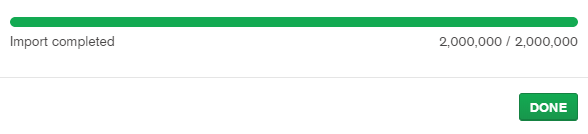
* On click of import the documents start getting imported



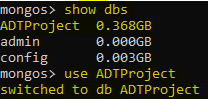
* Based on our experience trying to import 5 million records is a not a good idea on local machines as we kept running into the following error

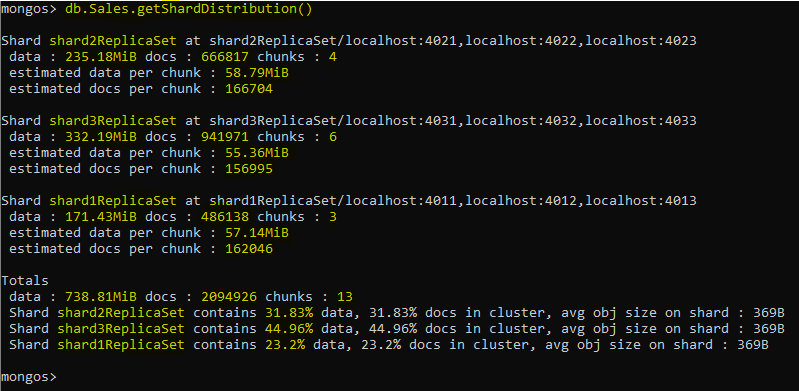


* This might be a limitation of the hardware of the machines and if running on the server, this problem might be mitigated
* Therefore, we reduced the number of documents to 2 million and the import completed without any issues



* As a consequence of having the routing server enabled and having the configuration server also enabled, each of these documents should have technically been routed to the shards to maintain even load
* We can validate the sharded nature of data by running the command
  + db.collectionName.getShardDistribution()
  + **Note:** We switch to use the “ADTProject” database by the command use ADTProject





* We can see the data size, number of documents and number of chunks at every shard mentioned
* The summary of overall percentage of documents at any shard is also mentioned
* This validates the sharding setup we have implemented.
* Any modification to the data on the primary replica set of any of the shards will be reflected on the secondary replica sets of the corresponding shards

**For existing Database and Collection**

We have seen above a from scratch setup of a sharded cluster and then population of a database. What if we have an existing database and collection? We follow the following steps

* Create an index on the field we wish to use as the shard key
* Enable sharding for the database using
  + sh.enableSharding(“database-name”)
* Add the collection to be sharded using
  + sh.shardCollection(“database-name.collection-name”, {“shard-key”: range or hash based sharding selected})
* **Note:** The shards have to be defined and configured as per requirement