**Sharding**

[Sharding](https://docs.mongodb.com/manual/reference/glossary/#term-sharding) is a method for distributing data across multiple machines.

There will be a Challenge to maintain huge data by a single server

By Implementing Sharding

**High Availabilty**

**Read and write Performance**

**Storage Capacity**

There are two methods for addressing system growth:

Vertical scaling

Horizontal scaling.

**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.

**Horizontal scaling:**

Horizontal Scaling involves dividing the dataset and load over multiple servers,

Additional servers to increase capacity as required.

Each machine handles a subset of the overall workload, potentially providing better efficiency than a single high-speed high-capacity server.

Only requires adding additional servers as needed, which can be a lower overall cost than high-end hardware for a single machine.

**Sharded Cluster**

A MongoDB [sharded cluster](https://docs.mongodb.com/manual/reference/glossary/" \l "term-sharded-cluster) consists of the following components:

* [**shard**](https://docs.mongodb.com/manual/core/sharded-cluster-shards/)**:** Each shard contains a subset of the sharded data. Each shard can be deployed as a [replica set](https://docs.mongodb.com/manual/reference/glossary/#term-replica-set).

**QueryRouter:** The **mongos** acts as a query router, providing an interface between client applications and the sharded cluster.

Query routers are basically mongo instances, directs operations to the appropriate shard. The query router processes and targets the operations to shards and then returns results to the clients. A sharded cluster can contain more than one query router to divide the client request load. A client sends requests to one query router. Generally, a sharded cluster have many query routers.

* [**config servers**](https://docs.mongodb.com/manual/core/sharded-cluster-config-servers/)**:** Config servers store metadata and configuration settings for the cluster. As of MongoDB 3.4, config servers must be deployed as a replica set (CSRS). This data contains a mapping of the cluster's data set to the shards.

## Shard Keys

To distribute the documents in a collection, MongoDB [partitions](https://docs.mongodb.com/manual/reference/glossary/#term-data-partition) the collection using the [shard key](https://docs.mongodb.com/manual/reference/glossary/#term-shard-key).

The [shard key](https://docs.mongodb.com/manual/reference/glossary/#term-shard-key) consists of an immutable field or fields that exist in every document in the target collection.

Need to choose shard key when sharding a collection.

The choice of shard key cannot be changed after sharding.

A sharded collection can have only one shard key.

To shard a non-empty collection, the collection must have an [index](https://docs.mongodb.com/manual/reference/glossary/#term-index) that starts with the shard key.

For empty collections, MongoDB creates the index if the collection does not already have an appropriate index for the specified shard key.

The choice of shard key affects the performance, efficiency, and scalability of a sharded cluster.

A cluster with the best possible hardware and infrastructure can be bottlenecked by the choice of shard key.

## Chunks

MongoDB partitions sharded data into [chunks](https://docs.mongodb.com/manual/reference/glossary/#term-chunk). Each chunk has an inclusive lower and exclusive upper range based on the [shard key](https://docs.mongodb.com/manual/reference/glossary/#term-shard-key).

MongoDB migrates [chunks](https://docs.mongodb.com/manual/reference/glossary/#term-chunk) across the [shards](https://docs.mongodb.com/manual/reference/glossary/#term-shard) in the [sharded cluster](https://docs.mongodb.com/manual/reference/glossary/" \l "term-sharded-cluster) using the [sharded cluster balancer](https://docs.mongodb.com/manual/core/sharding-balancer-administration/).

The balancer attempts to achieve an even balance of chunks across all shards in the cluster.

Auto sharding

Shard3

Shard2

Shard1

**Sharding Servers**

Replica Sets

**Mongos**

Applns

**Config Servers**

C1 mongod

C2 mongod

C3 mongod

**Steps**

**Create a new folder to maintain all ConfigServers and Shard Servers**

**Start the shards using mongod and --shardsvr**

Mongod –dbpath “e:\Sharding\sh1” –port 2001 --shardsvr

Mongod –dbpath “e:\Sharding\sh2” –port 2002 --shardsvr

Mongod –dbpath “e:\Sharding\sh3” –port 2003 --shardsvr

**Start the config Servers using mongod**

Mongod -–dbpath “e:\Sharding\cf1” –-port 2010 --configsvr –- replSet “rset”

Mongod -–dbpath “e:\Sharding\cf2” –-port 2011 --configsvr –- replSet “rset”

Mongod -–dbpath “e:\Sharding\cf3” –-port 2012 --configsvr –- replSet “rset”

**Open Mongo with port 2010 //Any one Config Servers port**

Mongo –port 2010

**Initiate the replicaSet**

Rs.initiate()

Rs.add(“rama:2011”)

Rs.add(“rama:2012”)

**U can check the no.of Replica servers rs.status()**

**Open Mongos as configdb with replica set**

Mongos –port 5000 –configdb rset/rama:2010,rama:2011,rama:2012

**Open Mongo using mongosPortNumber**

Mongo –port 5000

Will automatically starts mongos> prompt

**Add Shards**

Sh.addShard(“rama:2001”);

Sh.addShard(“rama:2002”);

Sh.addShard(“rama:2003”);

To check the status of the shards sh.status()

**Create a Database at Mongos>**

Use shardDb

**At mongos Enable sharding on the created Database**

Sh.enableSharding(“shardDb”)

**Add a ShardKey to the shard Collection**

Sh.shardCollection(“shardDb.myCollection”,{\_id:1})

**Using MongoBooster Connect to the Mongos Port Instance and populate the data in DB**

use shardDb

for(var i=1; i<100000; i++)

{

db.myCollection.insert({"value":i})

}

**Check the Status of distribution**

db.myCollection.getShardDistribution()

**To Remove the shard From the shardCluster**

Must use admin db ie admin previlages

use admin

db.runCommand({"removeShard":"shard0000"})

note : Data will be copied from the deleted shard to the remaining shards and distributed among them , once the copying completed , then data from Deltedshard will get removed

we can check the data distribution in remaining shards

db.myCollection.getShardDistribution()