Sharding

This lesson introduces the concept of Sharding, explains what shards are, what a sharding cluster is, and how they are used.

WE'LL COVER THE FOLLOWING

Limitations of Replica Sets

We learned that replica sets gave us the ability to hold data in multiple databases and thus, give us a certain level of fault tolerance and data duration.

However, this approach has certain limitations. As previously mentioned, all write operations go to the primary node, which makes it the bottleneck of the system. This means that if the system grows, the primary node will be overused and, eventually, it will be limited with hardware limitations like RAM, the number of CPUs, and disk space.

Scaling

There are two scaling methods used to address this issue:

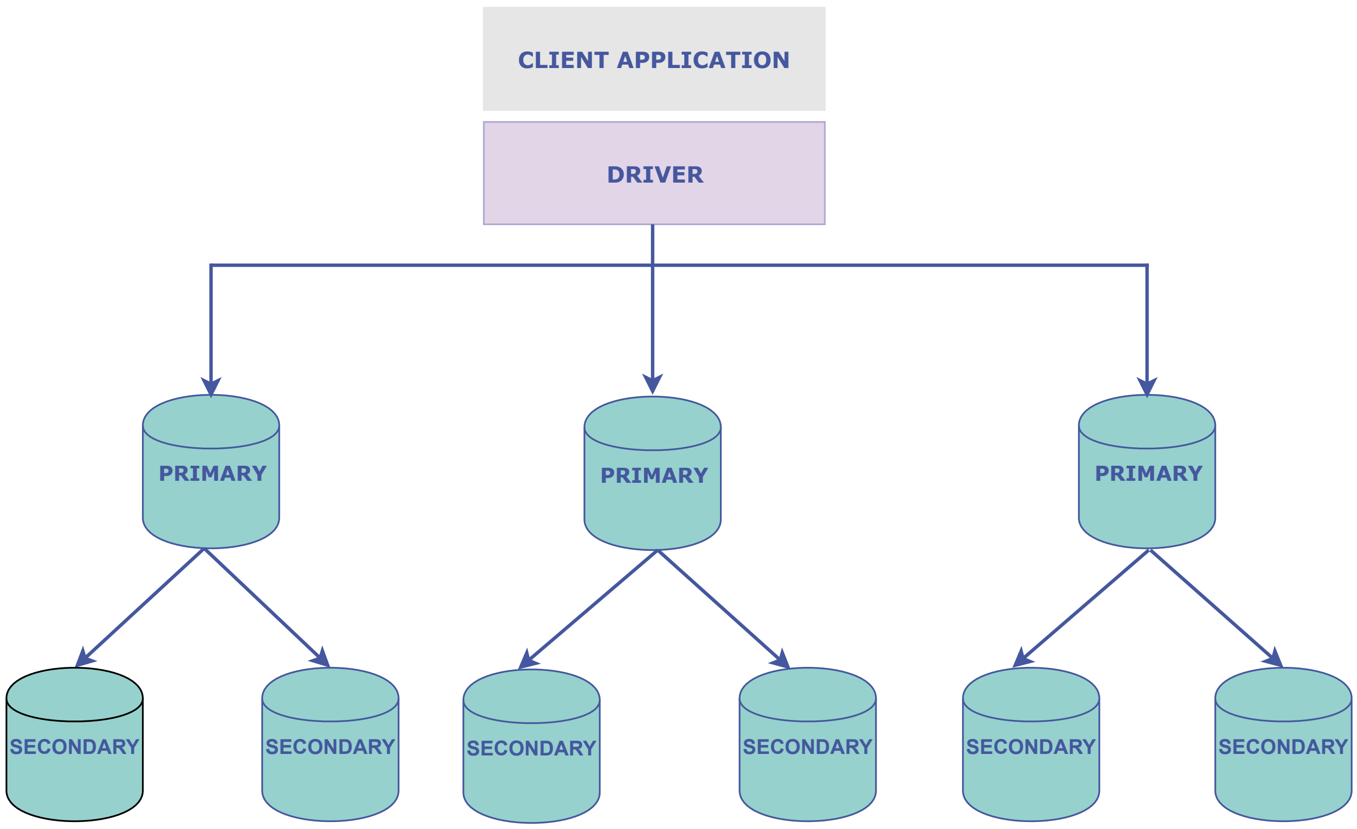
- **Vertical scaling** – This method involves increasing the performance of the server, on which the primary node is running, by adding more RAM or increasing the disk space. But this is very expensive and not the best solution for distributed systems.

- **Horizontal scaling** – This method includes dividing data and loading it onto multiple servers.

Sharding

Horizontal scaling in MongoDB is done by **sharding**.

The main goal of sharding is to provide us with something like this:



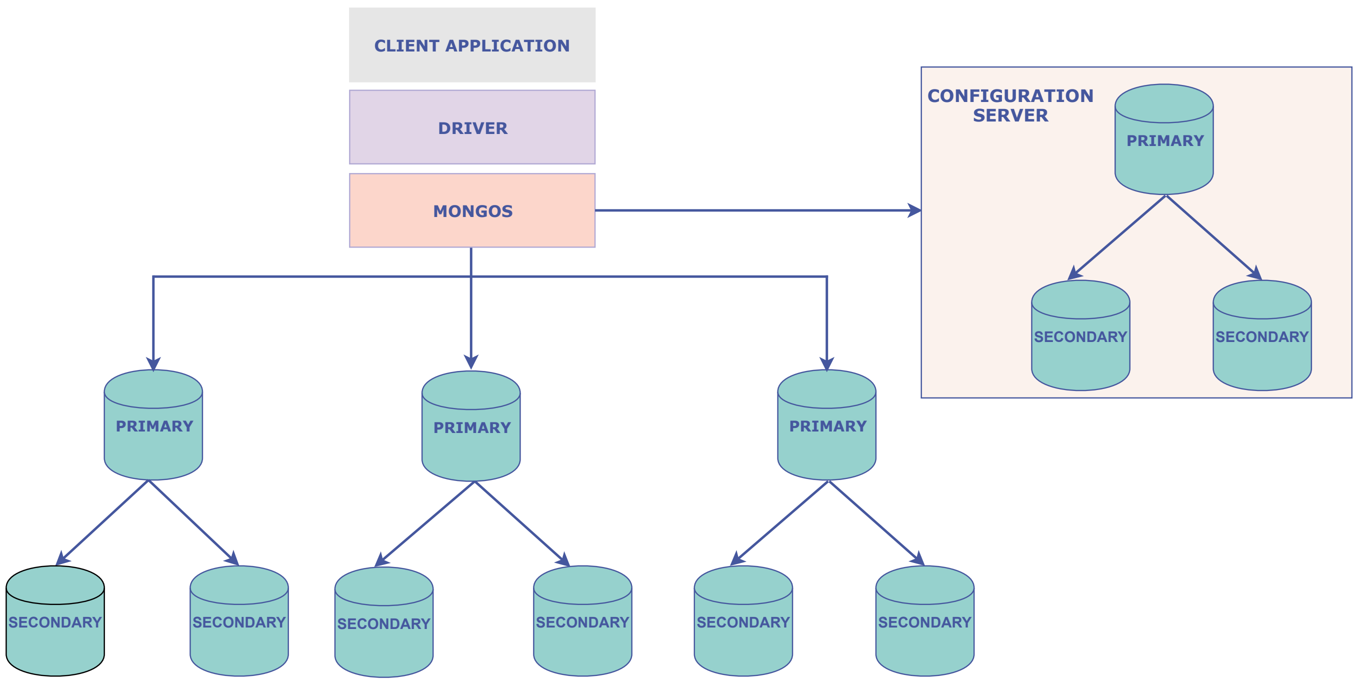
The idea is to have multiple replica sets, with multiple primaries, that will divide data and load it among themselves. Each of these replica sets is called a **shard**, but multiple shards are not enough to achieve the proper functionality of this kind of system.

Sharding Cluster

Firstly, there is a need for a router that will route queries and operations to the proper shard. For these purposes, MongoDB provides a new daemon process called **mongos**.

Also, this system needs to know which part of the data is in which shard; this is done by an additional replica set called **configuration server**.

The combination of *multiple shards*, *mongos processes*, and *configuration* *servers* is called a **sharding cluster**, and it looks something like this:



Sharding itself is done on a collection level.

This means that a defined collection is distributed among shards, not the whole database, and that is done by calling the method:

```
sh.shardCollection(“DATABASE_NAME.COLLECTION_NAME”, SHARD_KEY)
```

Once this command is run, the defined collection is distributed among different shards. Each shard will contain a range of data of a defined collection, and the *mongos* process will send queries to the proper shards.

Conclusion

Replica Sets and *Sharding* are important MongoDB features.

- By using replica sets, MongoDB is able to create recoverable and highly durable clusters.
- By using sharding, MongoDB is able to meet the demands of data growth and horizontal scaling.

In this way, MongoDB has been able to cover some of the major requests on the market today.

Now that you’ve become familiar with MongoDB basics we will learn how to use MongoDB in .NET. However, before that, let’s take a short quiz!