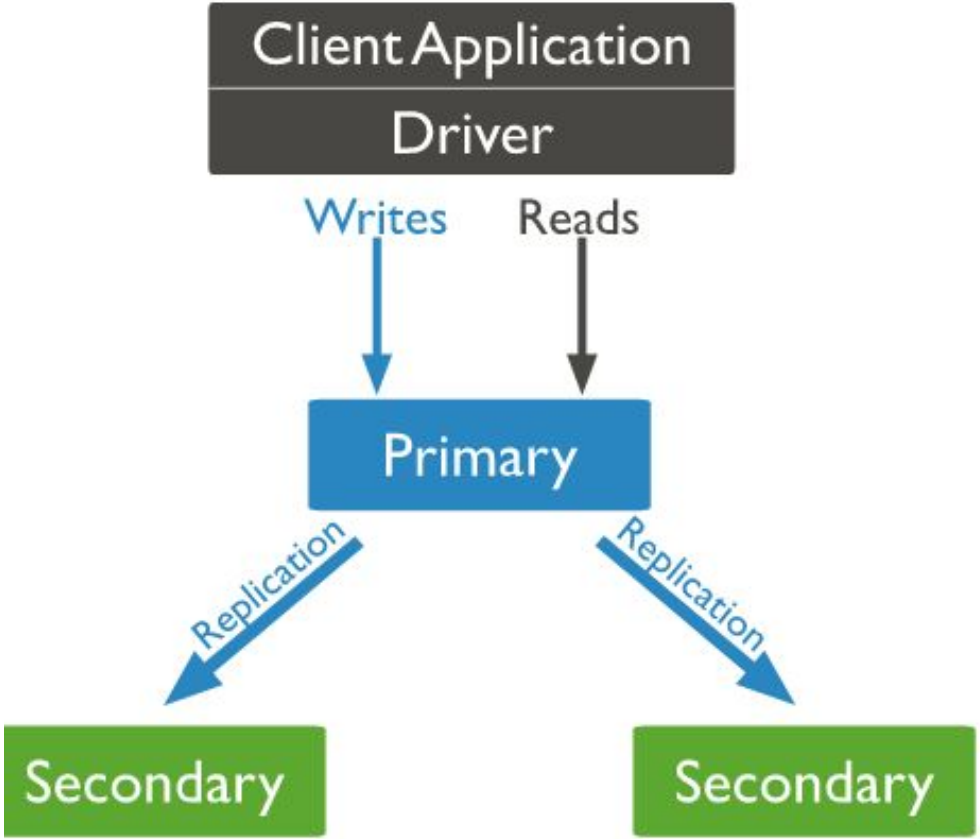

# Introduction to replication in mongo

# Replication

A replica set in **MongoDB** is a group of mongod processes that maintain the same data set. In simple terms replication is maintaining redundant copies of data on multiple machines.

Replica sets provide redundancy and high availability, and are the basis for all production deployments.


Client Application
Driver
Writes
Reads
Primary
Replication
Replication
Secondary
Secondary

# Advantages of Replication

- **High availability** of data in case of a failover.
- **Durability** of data which makes it lot more safer and easy recovery in case of a disaster.
- **Increased** read scalability as the reads can be performed even from the secondary database.
- In case of geographic workloads reads can be even more faster if secondary servers are present in different geographic locations.

Although replication provides these 4 advantages but the primary reasons why we do replication is due to **High availability** and **Durability of data.**

# Replica set Oplog

- The oplog (operations log) is a special capped collection that keeps a rolling record of all operations that modify the data stored in your databases. MongoDB applies database operations on the primary and then records the operations on the primary's oplog. The secondary members then copy and apply these operations in an asynchronous process. All replica set members contain a copy of the oplog, in the local.oplog.rs collection, which allows them to maintain the current state of the database.

# Some features in Replication

- **Asynchronous replication:**

  Replication in mongo happens asynchronously i.e. writes are done on primary for which the client gets the acknowledgement way before the writes are actually replicated on secondary servers.

- **Master/slave replication:**

  Mongo follows master/slave notation wherein only one node will handle all write request and will serve as the master of all available nodes, others will just do what the primary node already did.

# Some features in Replication (Contd.)

- **Statement based replication :**

  In statement based replication the replication is done via the write statements which are send to secondary nodes with the help of oplog collection.

  for eg:- we do a delete operation in primary i.e. db.emp.remove({age:30}), suppose it deletes 100 documents, so statement based replication will actually replicate by sending 100 statements each containing statement to remove document by _id field.

  Statement based help overcome issues which might come if 2 servers are running of different mongo version, if diff engine is used one saving uncompressed data and other using compressed data. Such issue might come if binary based replication was used.

# Some features in Replication (Cont.)

- **Automatic failover :**

  Mongo provides the feature of automatic failover wherein whenever primary node goes down one of the available nodes becomes the primary and whenever the failed node recovers it starts working as secondary node.

  Drivers are replica set aware, so if a failover occurs driver will hit all other members of replica sets till it identifies the primary node and than start talking to the same.

- **Automatic rollbacks in node recovery:**

  Whenever a node fails it becomes a secondary node on recovery. But what about the data that it wrote but was never able to replicate on secondary servers due to async replication ? In such cases mongo automatically rollbacks such writes and the server which is restored recently starts replicating from primary after the rollback is done.

# Setting up replication

- Set up a config object containing information of all members of replica set

```
Var cfg = {
            _id:’anand’ ,
             members : [
                        {_id:0, host:”localhost:27001”},
                        {_id:1, host:”localhost:27002”},
                        {_id:2, host:”localhost:27003”}
                        ]}
```

- Start the replica set by
  rs.initiate(cfg);

# Replica set commands

- rs.status() // gets the status of the replica set
- rs.initiate(cfg) // initiates a replica set with specified configuration object
- rs.conf() // it will show us config related details of replica set
- rs.stepDown(sec) // to take down primary for a certain period of time
- rs.reconfig(cfg) // to update the config of replica set
- rs.add(memberConfigObj) // adds a new host to replica set
- rs.remove(hostPortStr) // removes a host from replica set
- rs.freeze(secs) // make a node ineligible to become primary for specified duration
- rs.slaveOk() // to allow mongo server to run read commands on secondary
- rs.isMaster() // to check if current node is a primary node

# Read preferences

Following are the available read preferences of which by default is primary :

- primary
- primaryPreferred
- secondary
- secondaryPreferred
- nearest

# References

- https://university.mongodb.com/course/MongoDB/M102/2017_August/chapter/Chapter_4_Replication/lesson/5523404dd8ca3941fb77dfe9/tab/vertical_d04aef53b4c3

- https://docs.mongodb.com/manual/replication/

# Exercise

- Start MongoDB without authentication
- Create Database ttn and insert some data into user collection.
- Setup the replication with Artiber and verify that ttn database is replicated into secondary.

Add 2 more members to replica set with

rs.add();

Try adding a member which already have data.

# THANK YOU

# QUESTIONS ??