**Replication**

* Provides High availability and Disaster Recovery
* Increases the read capacity

the server with which application is connected is Primary Server and others are the secondary

Write will always happens from Primary Server

Application

Data in

doc Format

MongoD

Secondary Server

Only for voting purpose

Secondary server

OrbitrayrServer

* The communication between servers is sending heartbeats
* Secondary servers wait for 10 secs for the reply from Primary to response , if no response in 10 secs the secondary Servers will decide that primary was crashed .
* If primary unavailable , then one of the secondary become Primary after election. We can optinally add orbitor server to resolve deadlock storing
* Election happened between Secondary servers and basing on amount, freshness of data, and configuration ,a secondary will become primary

Replica set(the server machines) holds the same dataset

One momgod which is primary receives

* Primary saves the changes and log the the changes in uplog . uplog will be sent to secondary and processed and saved at secondary

**Steps**

Create a folder Replication and create three folders m1,m2,m3

Start three mongoD process by using

Mongod -–dbpath “Replication/m1” – -replSet “RsName” – -port 2003

Mongod -–dbpath “Replication/m2” – -replSet “RsName” – -port 2004

Mongod -–dbpath “Replication/m3” – -replSet “RsName” – -port 2005

Next connect with any of the machine start the replication process

Open three MongoShells with three ports

In any one of the mongoshell use rs.initiate() will initiates as Primary

Rs.config() – to check how many servers are there in the replica set

Add secondary

Rs.add(“Computername:2004”)

Rs.add(“Computername:2005”)

To check

Rs.status()

Now add some data in the primary server

Now open any ine secondary server

Rs.slaveOk() – making secondary slave ready to read the data

Fire a query for checking whether the data is available or not

Same thing can be done on another secondary server too