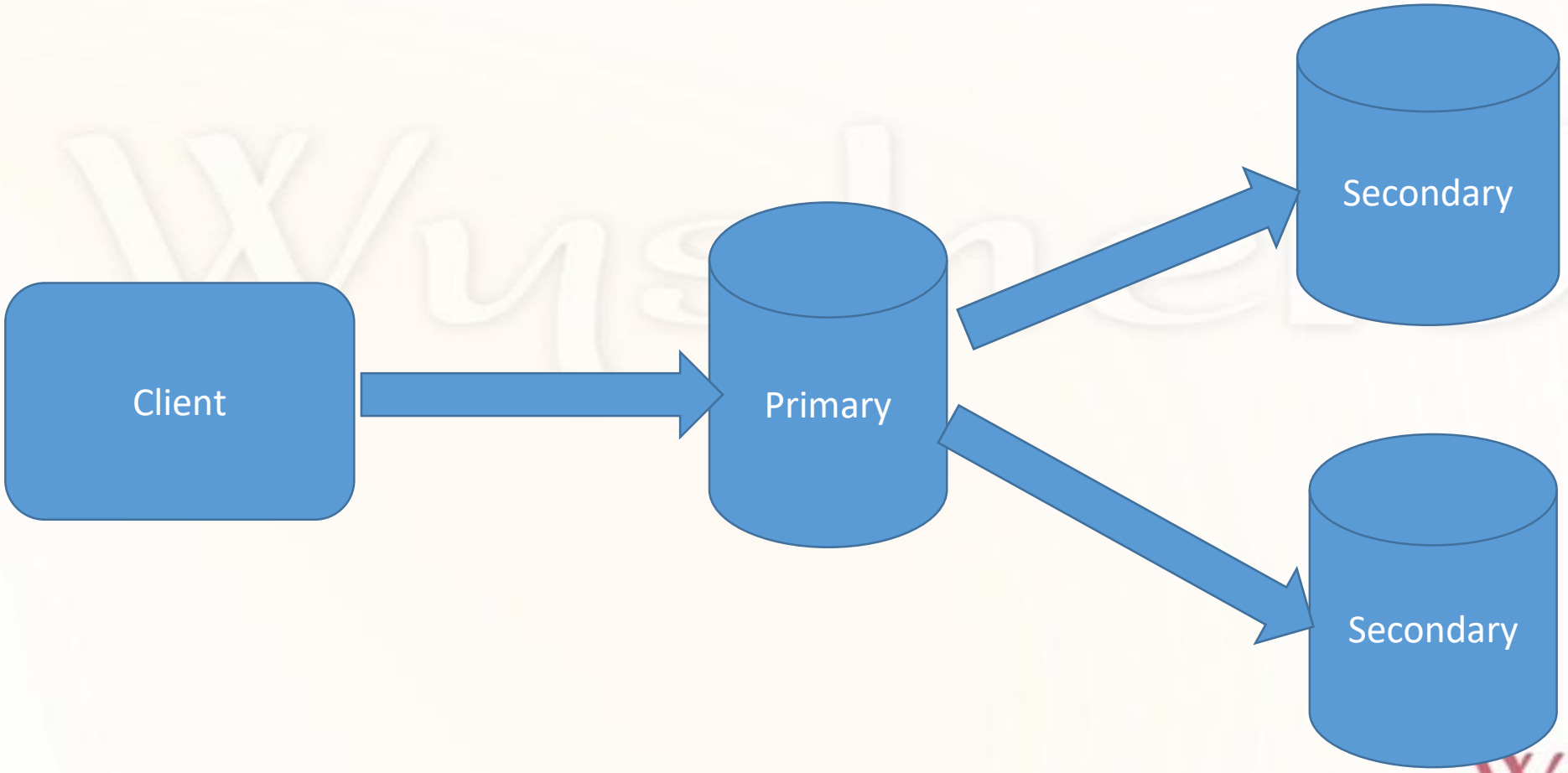
Replication

What is replication

- Replication is a way of keeping identical copies of your data on multiple servers
- With MongoDB, you set up replication by creating a replica set
- A replica set is a group of servers with one primary, the server taking writes, and multiple secondaries, servers that keep copies of the primary's data
- If the primary crashes, the secondaries can elect a new primary from amongst themselves.
- MongoDB Atlas provides replicasets for test, dev and production

Important points to consider



Important points to consider

- The members of the replica set should be started with ip other than local ip.Use –bind_ip option
- Enable authentication and authorization
- rs is a global variable , containg methods to administrate replicat sets
- Replica sets are limited to 50 members in total and only 7 voting members
- If you are creating a replica set that has more than seven members, every additional member must be given zero votes
- > rs.add({"_id" : 7, "host" : “10.17.57.10:27025", "votes" : 0})

Setting up replication

- In this configuration , we have 3 servers. For simiplicity we are starting all the three servers from the same VM, but on three different ports
- Server lists
    - 10.17.57.10 27018
    - 10.17.57.10 27019
    - 10.17.57.10 27020
- Create all the directories , required for data directory and log file
    - $ mkdir /u01/data1 /u01/data2/ /u01/data3

Setting up replication

- Start all the three mongod process
  - mongod --bind_ip localhost,10.17.57.10 --replSet rs0 --dbpath /u01/data1 --port 27018 --oplogSize 200 --logpath /u01/data1/mongo.log --fork
  - mongod --bind_ip localhost,10.17.57.10 --replSet rs0 --dbpath /u01/data2 --port 27019 --oplogSize 200 --logpath /u01/data2/mongo.log --fork
  - mongod --bind_ip localhost,10.17.57.10 --replSet rs0 --dbpath /u01/data3 --port 27020 --oplogSize 200 --logpath /u01/data3/mongo.log --fork

Setting up replication

- Connect to one of the mongod process, and create a variable with member information and replica set name . Make sure to use the replica set name used while starting up the mongod servers

```
rconfig={
_id:"rs0",
members:[
{_id:0,host:"10.17.57.10:27018"},
{_id:1, host:"10.17.57.10:27019"},
{_id:2, host:"10.17.57.10:27020"}
]
}
```

- Initiate the replication configuration
  - > rs.initiate(rconfig)

Setting up replication

- Replica set will elect a primary soon
- The mongo shell prompt displays if its primary or secondary
- > rs.status() will show the status of the replication configuration
- rs0:PRIMARY> indicates , you are connected to the primary server of “rs0”, replication configuration
- > db.isMaster() helps to determine if we have connection to the primary . Mostly used in the programing
- Both commands show the list of all the members of the replica set

Reading from the secondary

- By default reading from secondary is not enabled.
  - "errmsg" : "not master and slaveOk=false",
- To solve
  - Connect to secondary
  - > rs.secondaryOk()
- Try to select documents which were inserted on primary
  - > use movies
  - > db.movies.find()

Testing the failover secondary

- Connect to primary and shut it down
  - use admin
  - > db.shutdownServer()
- Connect to any other secondary
  - > rs.status()
  - Look for “"stateStr" : "PRIMARY“”, to understand which is the current primary

Adding & Removing members

- Start a new mongod
- mongod --bind_ip localhost,10.17.57.10 --replSet rs0 --dbpath /u01/data4 --port 27021 --oplogSize 200 --logpath /u01/data4/mongo.log –fork
- Connect to primary
- Add the new member
  - > rs.add("10.17.57.10:27021")
- Check the status of the configuration
  - > rs.config()
- Remove a member
  - > rs.remove("10.17.57.10:27021")

Reconfiguring Replicaset

- Lets assume, that one of the member is starting on a new port. To solve thi , we have to change the port number of that member.
- Stop the member , and restart using the new port number
  - lastHeartbeatMessage" : "Error connecting to 10.17.57.10:27021 :: caused by :: Connection refused", // rs.status displays this message
- Modify the member with new port number
  - > var conf=rs.config()
  - > conf.members[3].host="10.17.57.10:27022“
- Reconfigure the replica set
  - > rs.reconfig(conf)
- Check the configuration and status
  - > rs.config()
  - > rs.status()
  -

Considerations while designing replicaset

- Design configuration such a way that primary can reach more than half of the members
- Majority is designed by the number of members in the configuration , not based on the available members.
- Members will ping other members every 2 second.
- If a heartbeat does not return from a member within 10 seconds, the other members mark the delinquent member as inaccessible.
- If the member unable to reach the primary in specified time, it will propose itself as primary
- Other members after receiving the new primary request checks the following.
  - Are they able to reach old primary
  - Is the member who request to primary has the latest transaction
  - Priority of the member

States that indicate a problem in replication

- DOWN
  - If a member was up but then becomes unreachable
- UNKNOWN
  - If a member has never been able to reach another member, it will not know what state it's in, so it will report it as UNKNOWN.
- REMOVED
  - This is the state of a member that has been removed from the set.
- ROLLBACK
  - This state is used when a member is rolling back data,

Different States in Replication

- STARTUP
  - This is the state a member is in when it's first started, while MongoDB is attempting to load its replica set configuration. Once the configuration has been loaded, it transitions to STARTUP2.
- STARTUP2
  - This state lasts throughout the initial sync process, which typically takes just a few seconds. The member forks off a couple of threads to handle replication and elections and then transitions into the next state: RECOVERING.
- RECOVERING
  - This state indicates that the member is operating correctly but is not available for reads
- ARBITER
  - Arbiters have a special state and should always be in this state during normal operation.

Increase the size of Ops log

- Connect to the replica set member whose Ops log size to be increased.
- Verify the current size of the oplog:
  - > use local
  - > db.oplog.rs.stats(1024*1024).maxSize
- Change the oplog size of the replica set member:
  - > db.adminCommand({replSetResizeOplog: 1, size: 16000})