



# Replication and Replica Sets

Jason Zucchetto

*Consulting Engineer, 10gen*

# Agenda

- Replica Sets Lifecycle
- Developing with Replica Sets
- Operational Considerations
- Behind the Curtain

# Why Replication?

- How many have faced node failures?
- How many have been woken up from sleep to do a fail-over(s)?
- How many have experienced issues due to network latency?
- Different uses for data
  - Normal processing
  - Simple analytics

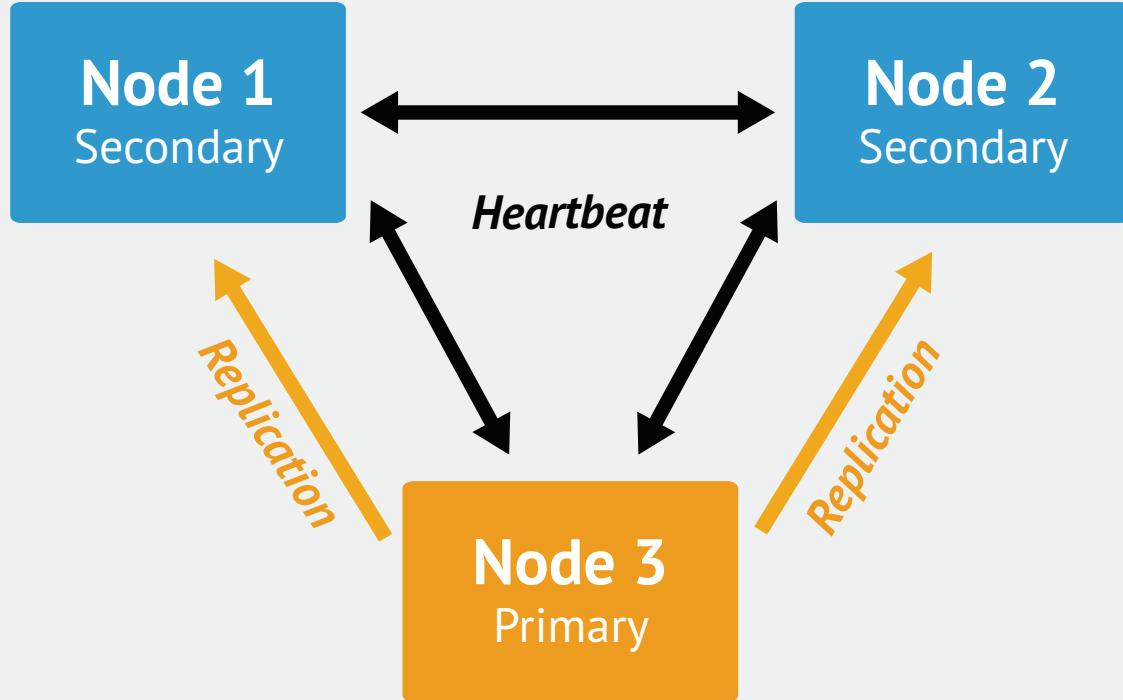
# ReplicaSet Lifecycle

Node 1

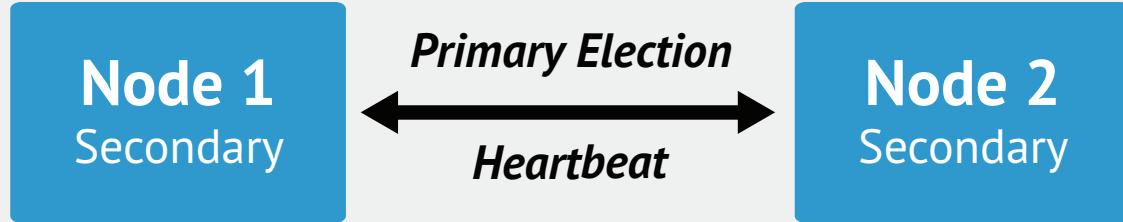
Node 2

Node 3

# Replica Set – Creation



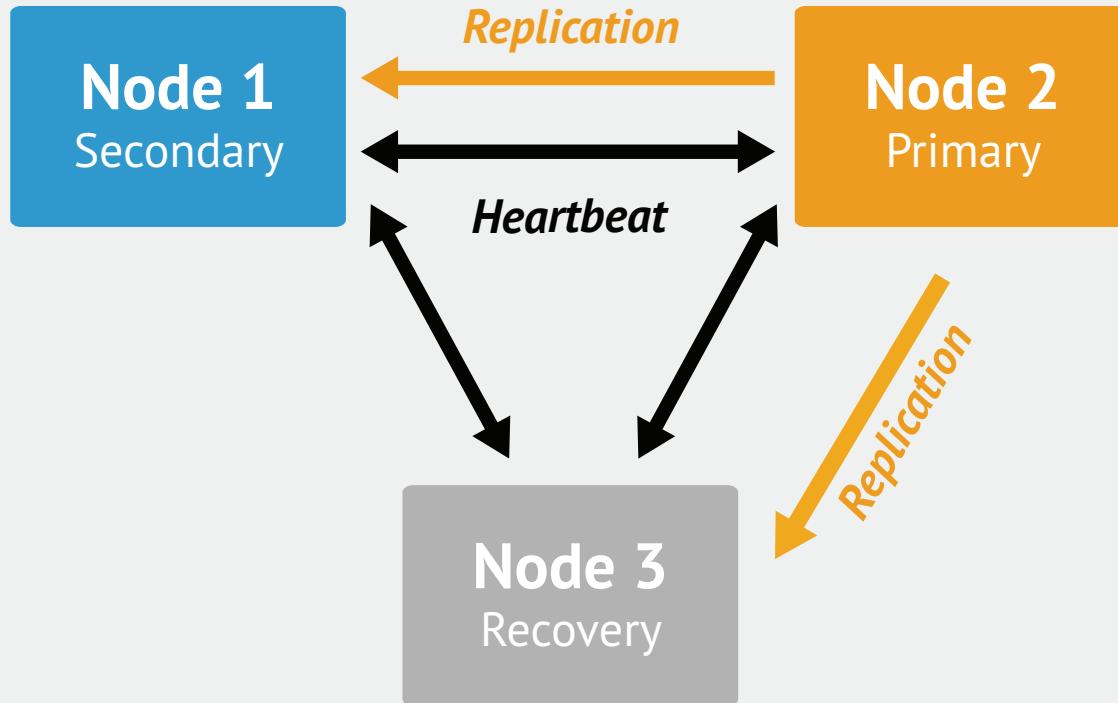
# Replica Set – Initialize



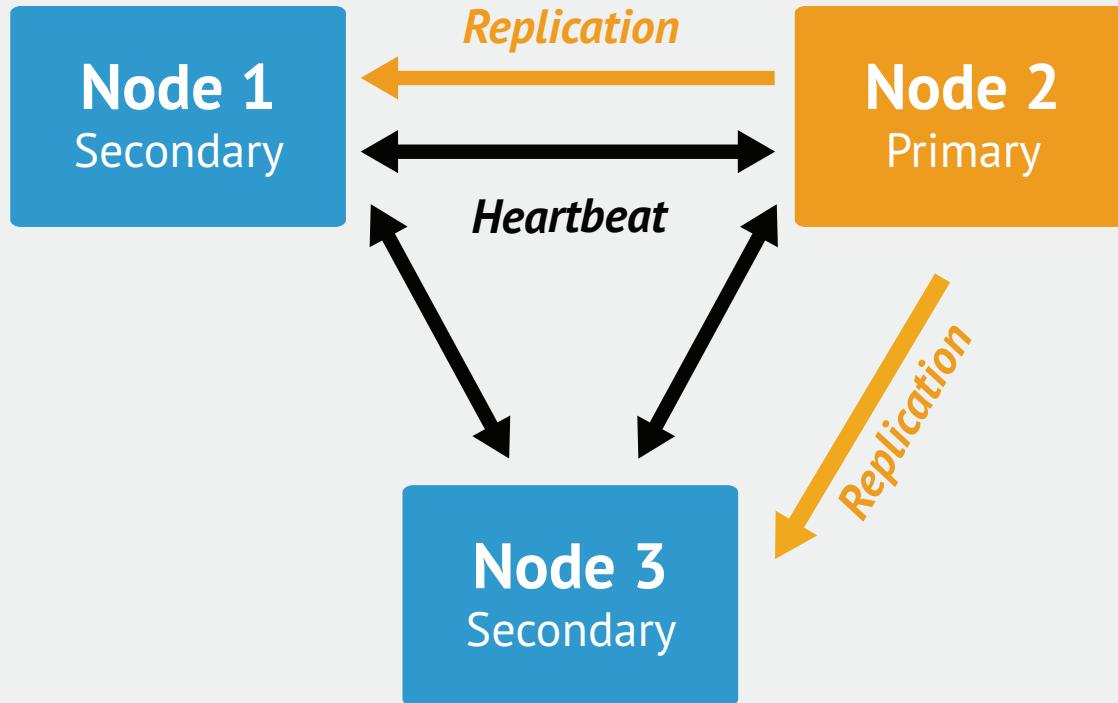
# Replica Set – Failure



# Replica Set – Failover

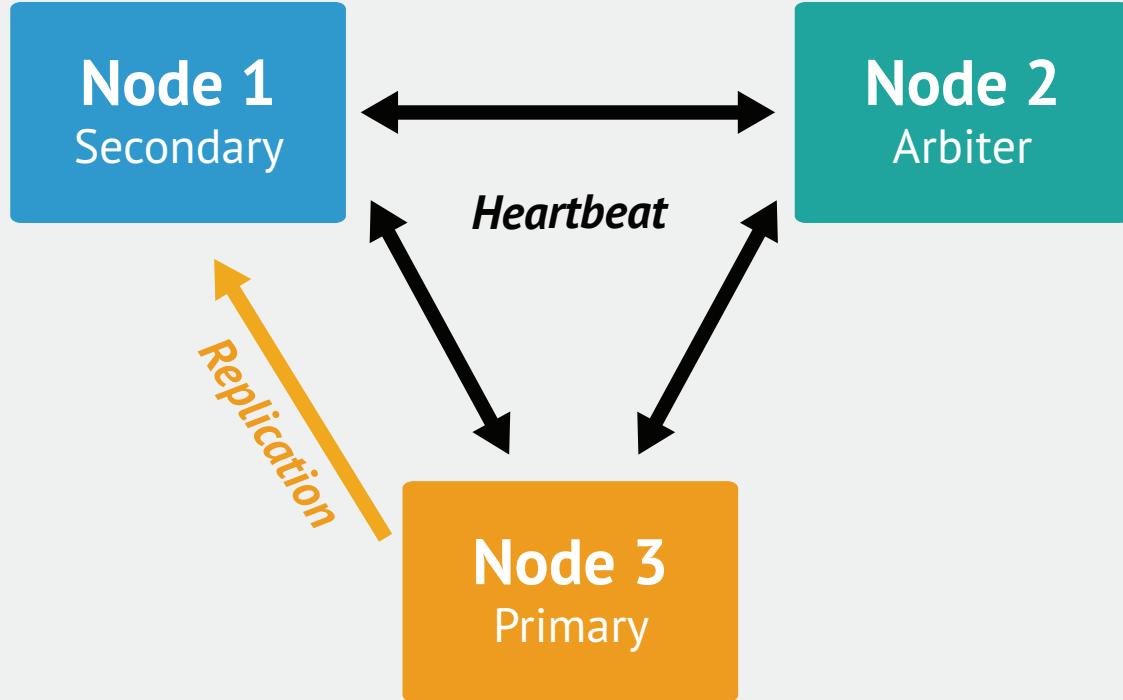


# Replica Set – Recovery



# Replica Set – Recovered

# ReplicaSet Roles & Configuration



# Replica Set Roles

# Configuration Options

```
> conf = {  
    _id : "mySet",  
    members : [  
        {_id : 0, host : "A", priority : 3},  
        {_id : 1, host : "B", priority : 2},  
        {_id : 2, host : "C"},  
        {_id : 3, host : "D", hidden : true},  
        {_id : 4, host : "E", hidden : true, slaveDelay : 3600}  
    ]  
}  
  
> rs.initiate(conf)
```

# Configuration Options

```
> conf = {  
  _id : "mySet",  
  members : [  
    {_id : 0, host : "A", priority : 3},  
    {_id : 1, host : "B", priority : 2},  
    {_id : 2, host : "C"},  
    {_id : 3, host : "D", hidden : true},  
    {_id : 4, host : "E", hidden : true, slaveDelay : 3600}  
  ]  
}  
  
> rs.initiate(conf)
```

Primary DC ↓

# Configuration Options

```
> conf = {  
  _id : "mySet",  
  members : [  
    {_id : 0, host : "A", priority : 3},  
    {_id : 1, host : "B", priority : 2},  
    {_id : 2, host : "C"},  
    {_id : 3, host : "D", hidden : true},  
    {_id : 4, host : "E", hidden : true, slaveDelay : 3600}  
  ]  
}  
  
> rs.initiate(conf)
```

Secondary DC  
Default Priority = 1



# Configuration Options

```
> conf = {  
  _id : "mySet",  
  members : [  
    {_id : 0, host : "A", priority : 3},  
    {_id : 1, host : "B", priority : 2},  
    {_id : 2, host : "C"},  
    {_id : 3, host : "D", hidden : true},  
    {_id : 4, host : "E", hidden : true, slaveDelay : 3600}  
  ]  
}  
  
> rs.initiate(conf)
```



Analytics  
node

# Configuration Options

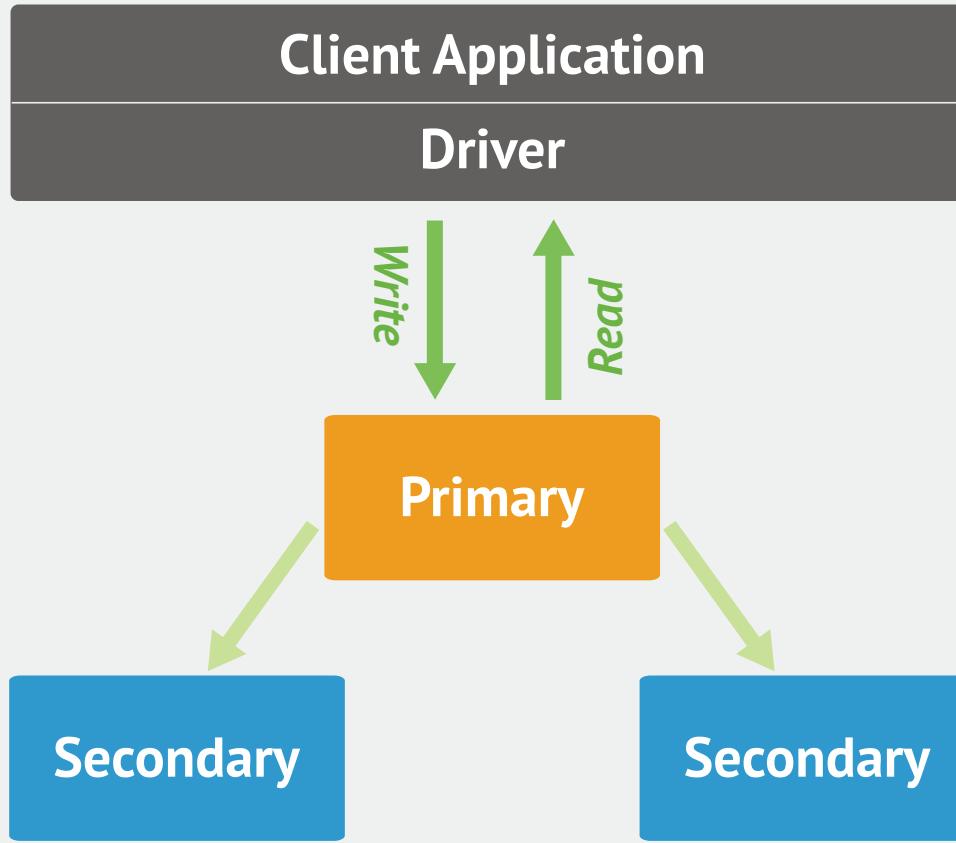
```
> conf = {  
  _id : "mySet",  
  members : [  
    {_id : 0, host : "A", priority : 3},  
    {_id : 1, host : "B", priority : 2},  
    {_id : 2, host : "C"},  
    {_id : 3, host : "D", hidden : true},  
    {_id : 4, host : "E", hidden : true, slaveDelay : 3600}  
]
```

```
}
```

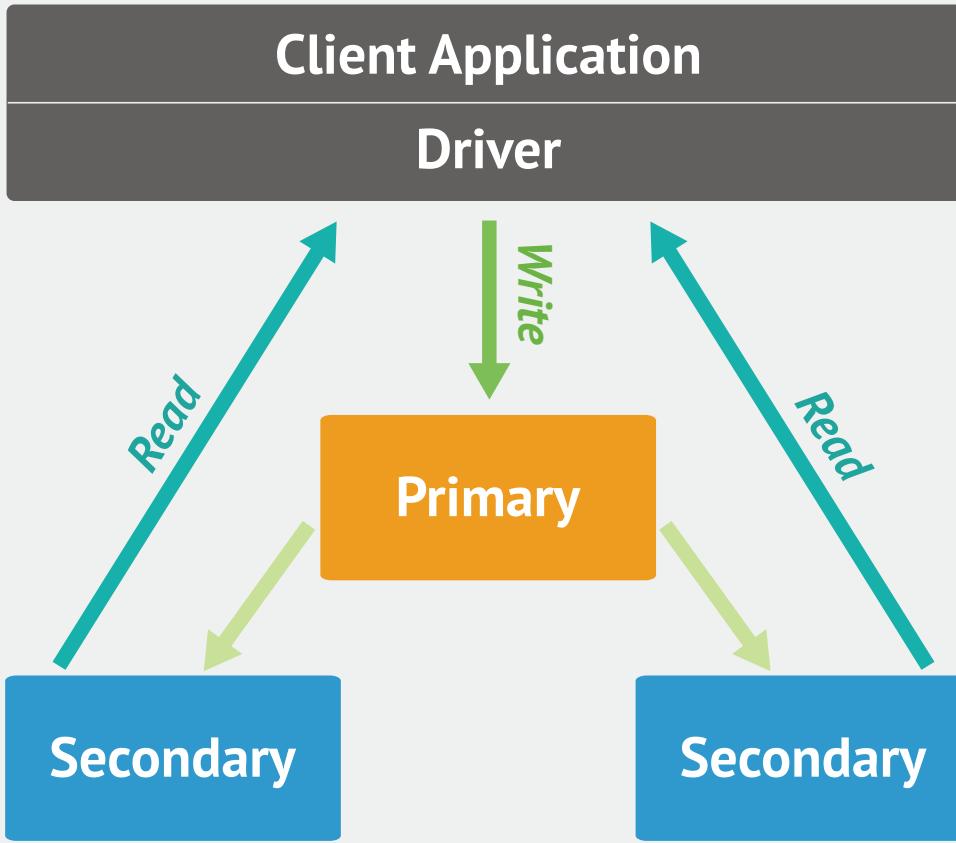
```
> rs.initiate(conf)
```

Backup node 

# Developing with Replica Sets



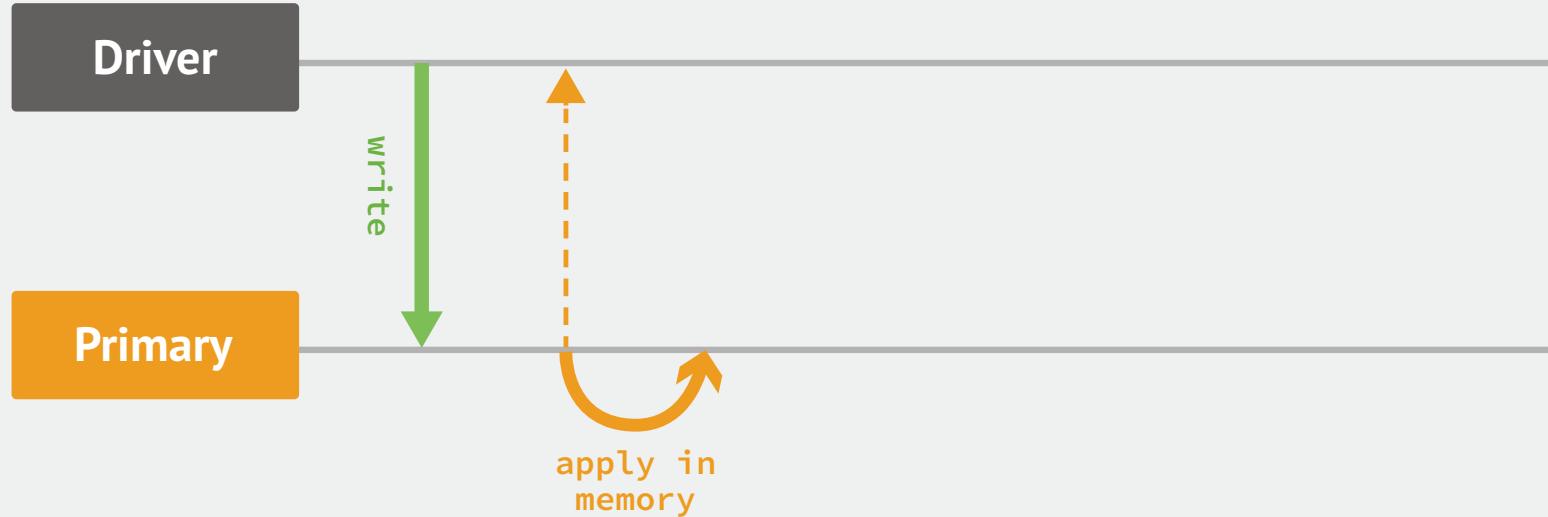
# Strong Consistency



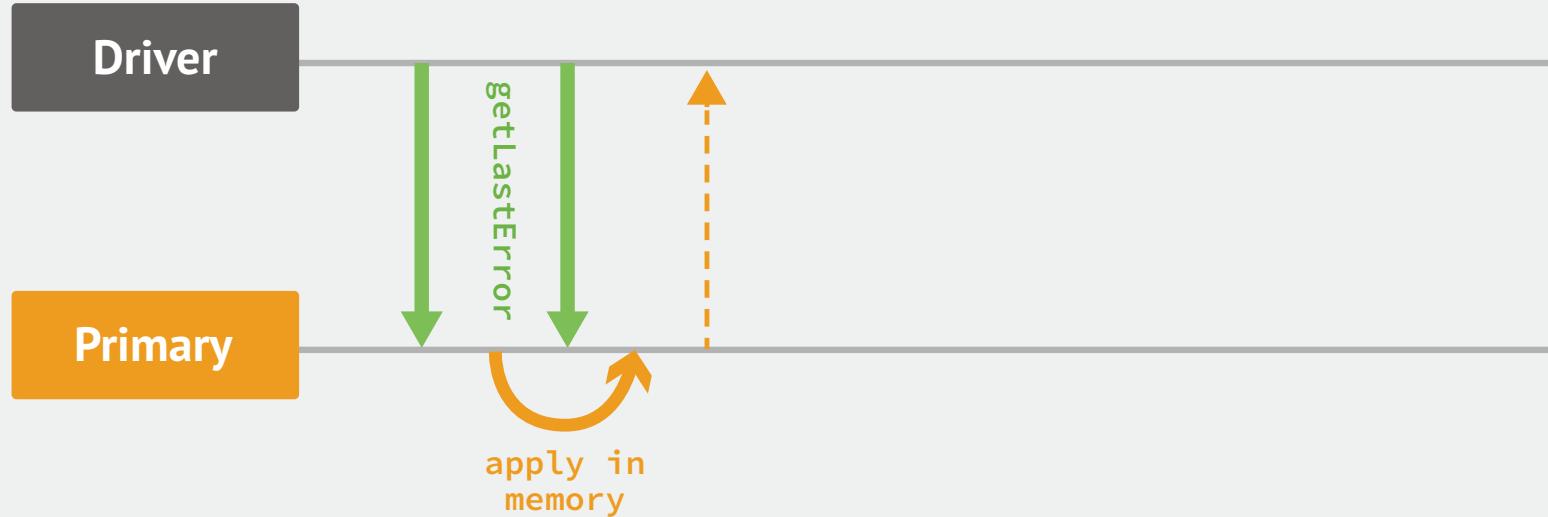
# Delayed Consistency

# Write Preference

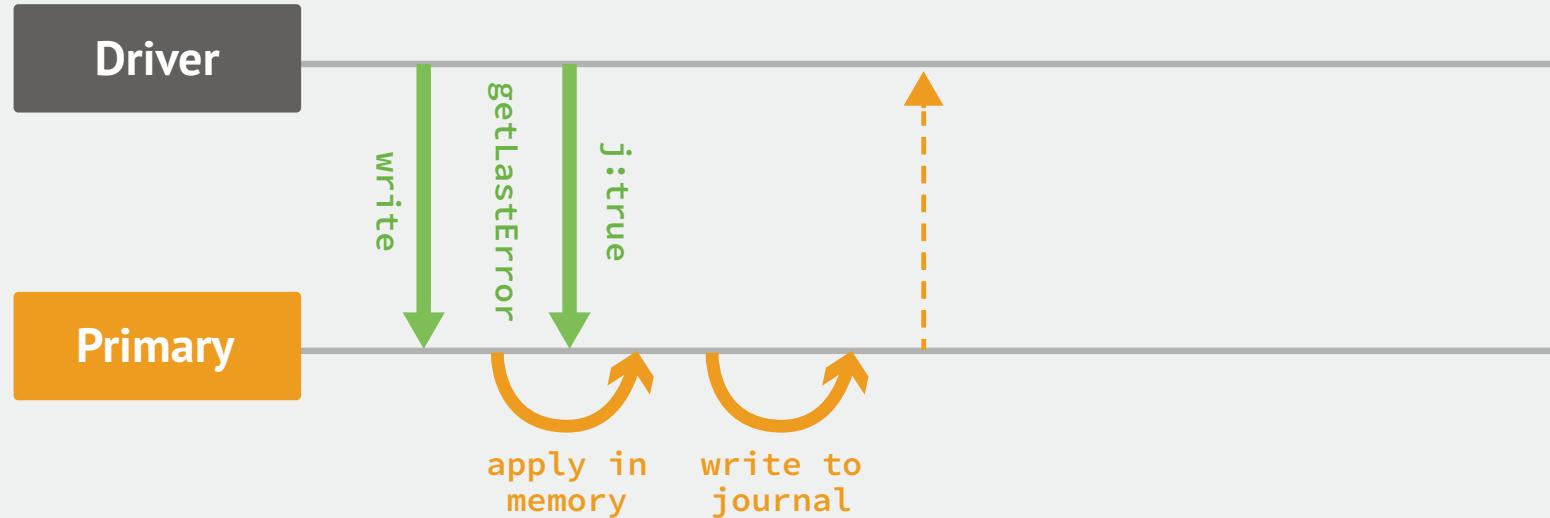
- Network acknowledgement
- Wait for error
- Wait for journal sync
- Wait for replication



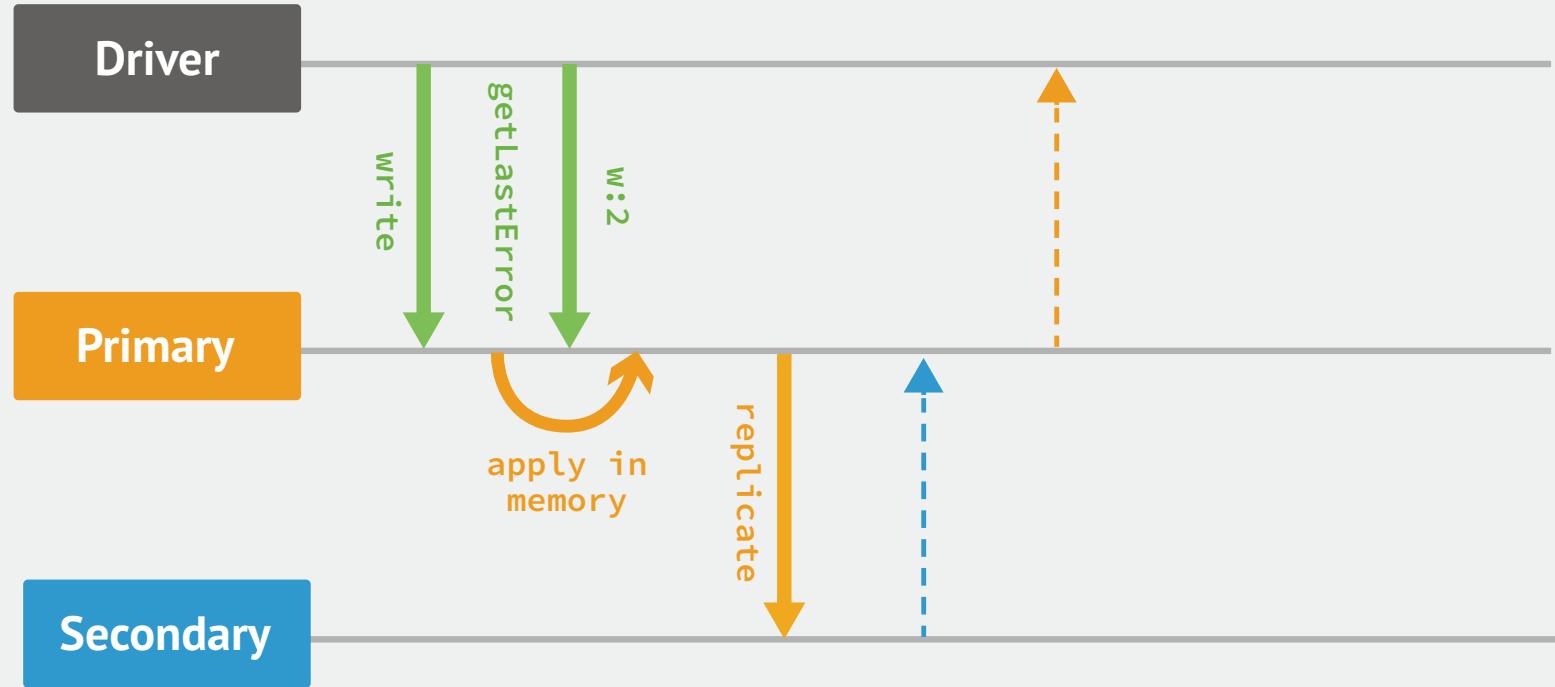
# Unacknowledged



# MongoDB Acknowledged (wait for error)



# Wait for Journal Sync



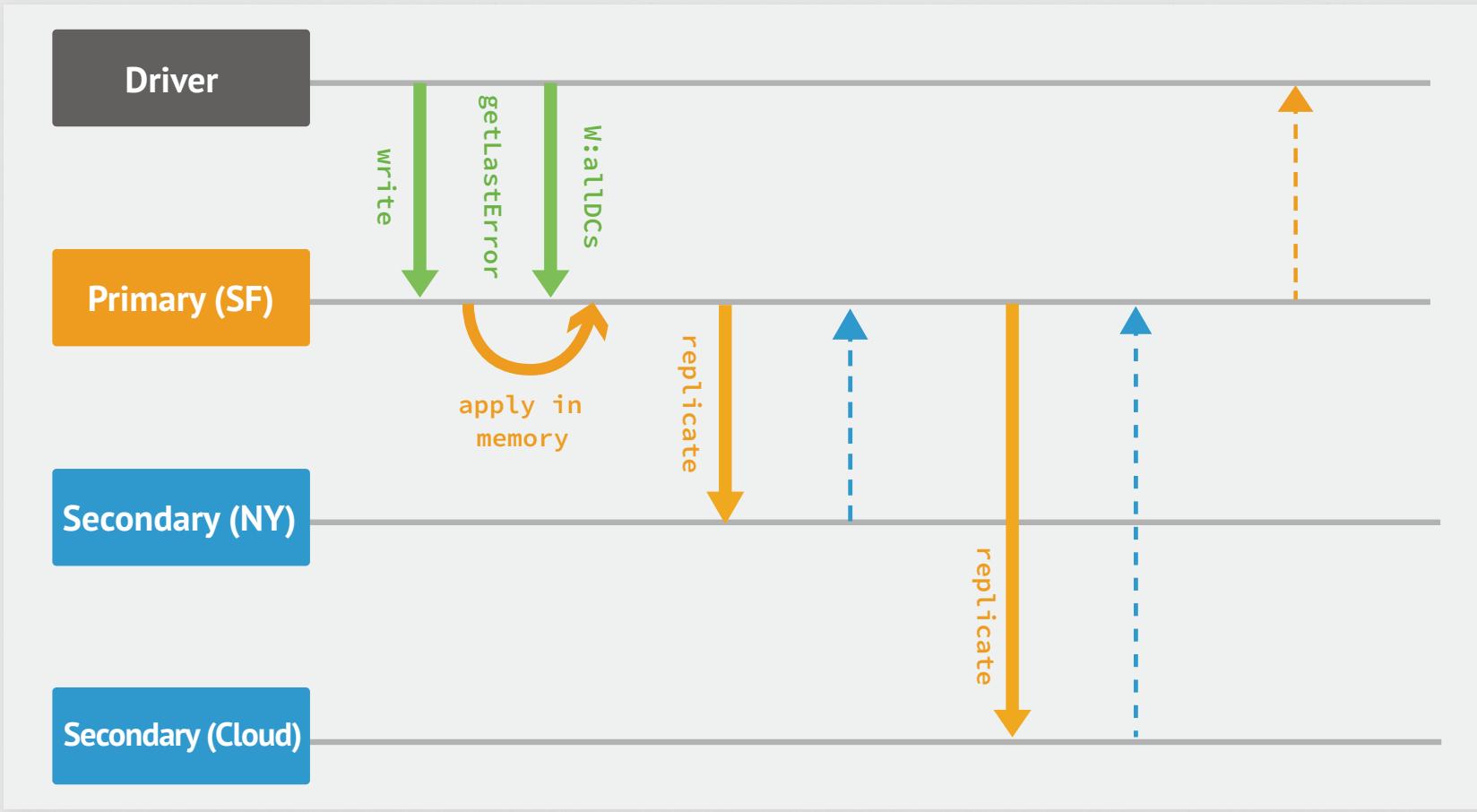
# Wait for Replication

# Tagging

- New in 2.0.0
- Control where data is written to, and read from
- Each member can have one or more tags
  - tags: {dc: "ny"}
  - tags: {dc: "ny",  
  subnet: "192.168",  
  rack: "row3rk7"}
- Replica set defines rules for write concerns
- Rules can change without changing app code

# Tagging Example

```
{  
  _id : "mySet",  
  members : [  
    {_id : 0, host : "A", tags : {"dc": "ny"}},  
    {_id : 1, host : "B", tags : {"dc": "ny"}},  
    {_id : 2, host : "C", tags : {"dc": "sf"}},  
    {_id : 3, host : "D", tags : {"dc": "sf"}},  
    {_id : 4, host : "E", tags : {"dc": "cloud"}}],  
  settings : {  
    getLastErrorModes : {  
      allDCs : {"dc" : 3},  
      someDCs : {"dc" : 2}} }  
}  
> db.blogs.insert({...})  
> db.runCommand({getLastError : 1, w : "someDCs"})
```



# Wait for Replication (Tagging)

# Read Preference Modes

- 5 modes (new in 2.2)
  - primary (only) - Default
  - primaryPreferred
  - secondary
  - secondaryPreferred
  - Nearest

When more than one node is possible, closest node is used for reads (all modes but primary)

# Tagged Read Preference

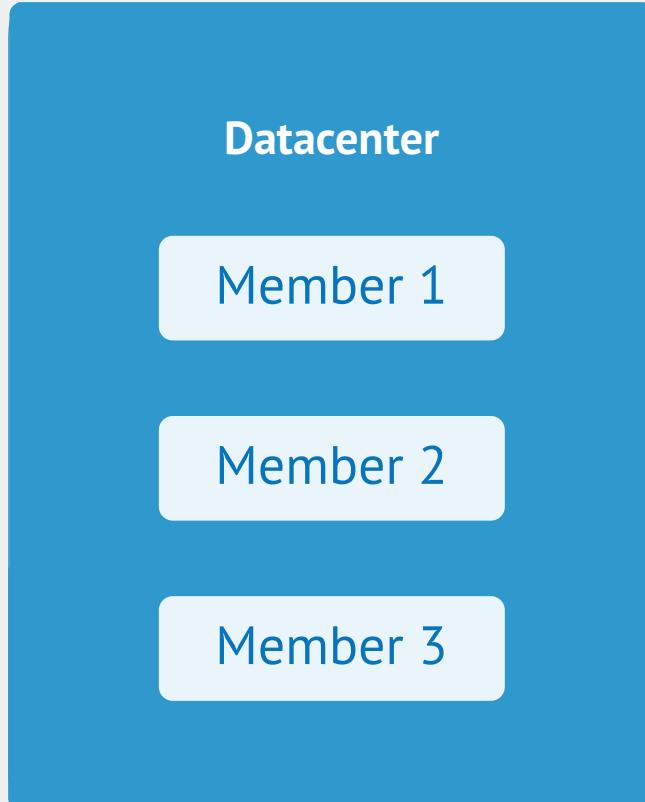
- Custom read preferences
- Control where you read from by (node) tags
  - E.g. { "disk": "ssd", "use": "reporting" }
- Use in conjunction with standard read preferences
  - Except primary

# Operational Considerations

# Maintenance and Upgrade

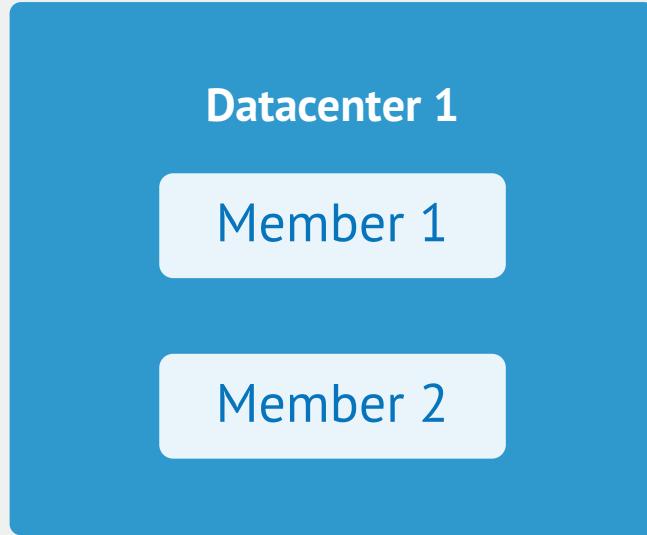
- No downtime
- Rolling upgrade/maintenance
  - Start with Secondary
  - Primary last

# Replica Set – 1 Data Center



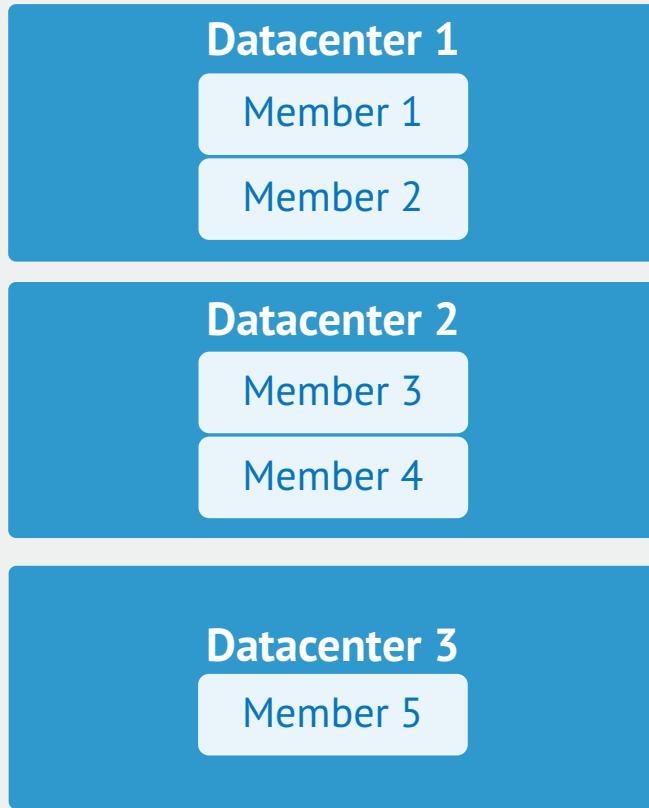
- Single datacenter
- Single switch & power
- Points of failure:
  - Power
  - Network
  - Data center
  - Two node failure
- Automatic recovery of single node crash

# Replica Set – 2 Data Centers



- Multi data center
- DR node for safety
- Can't do multi data center durable write safely since only 1 node in distant DC

# Replica Set – 3 Data Centers



- Three data centers
- Can survive full data center loss
- Can do  $w = \{ dc : 2 \}$  to guarantee write in 2 data centers (with tags)

# Behind the Curtain

# Implementation details

- Heartbeat every 2 seconds
  - Times out in 10 seconds
- Local DB (not replicated)
  - system.replset
  - oplog.rs
    - Capped collection
    - Idempotent version of operation stored

# Op(erations) Log is idempotent

```
> db.replsettest.insert({_id:1,value:1})  
{ "ts" : Timestamp(1350539727000, 1), "h" :  
  NumberLong("6375186941486301201"), "op" : "i", "ns" :  
  "test.replsettest", "o" : { "_id" : 1, "value" : 1 } }
```

```
> db.replsettest.update({_id:1},{$inc:{value:10}})  
{ "ts" : Timestamp(1350539786000, 1), "h" :  
  NumberLong("5484673652472424968"), "op" : "u", "ns" :  
  "test.replsettest", "o2" : { "_id" : 1 },  
  "o" : { "$set" : { "value" : 11 } } }
```

# Single operation can have many entries

```
> db.replsettest.update({}, {$set:{name : "foo"}}, false, true)

{ "ts" : Timestamp(1350540395000, 1), "h" :
NumberLong("-4727576249368135876"), "op" : "u", "ns" :
"test.replsettest", "o2" : { "_id" : 2 }, "o" : { "$set" : { "name" :
"foo" } } }

{ "ts" : Timestamp(1350540395000, 2), "h" :
NumberLong("-7292949613259260138"), "op" : "u", "ns" :
"test.replsettest", "o2" : { "_id" : 3 }, "o" : { "$set" : { "name" :
"foo" } } }

{ "ts" : Timestamp(1350540395000, 3), "h" :
NumberLong("-1888768148831990635"), "op" : "u", "ns" :
"test.replsettest", "o2" : { "_id" : 1 }, "o" : { "$set" : { "name" :
"foo" } } }
```

# What's New in 2.2

- Read preference support with sharding
  - Drivers too
- Improved replication over WAN/high-latency networks
- rs.syncFrom command
- buildIndexes setting
- replIndexPrefetch setting

# Just Use It

- Use replica sets
- Easy to setup
  - Try on a single machine
- Check doc page for RS tutorials
  - <http://docs.mongodb.org/manual/replication/#tutorials>



# Thank You

Jason Zucchetto

*Consulting Engineer, 10gen*