#codemash

MongoDB – Zero to Sharding

Part 3.1 : Replication

Sridhar Nanjundeswaran

Engineer, MongoDB Inc.

@snanjund

mongoDB

Agenda

- Introduction to Replica Sets
- Developing with Replica Sets
- Operational Considerations

- https://github.com/sridharn/codemash_2014/tree/master/replication

Introduction to Replica Sets

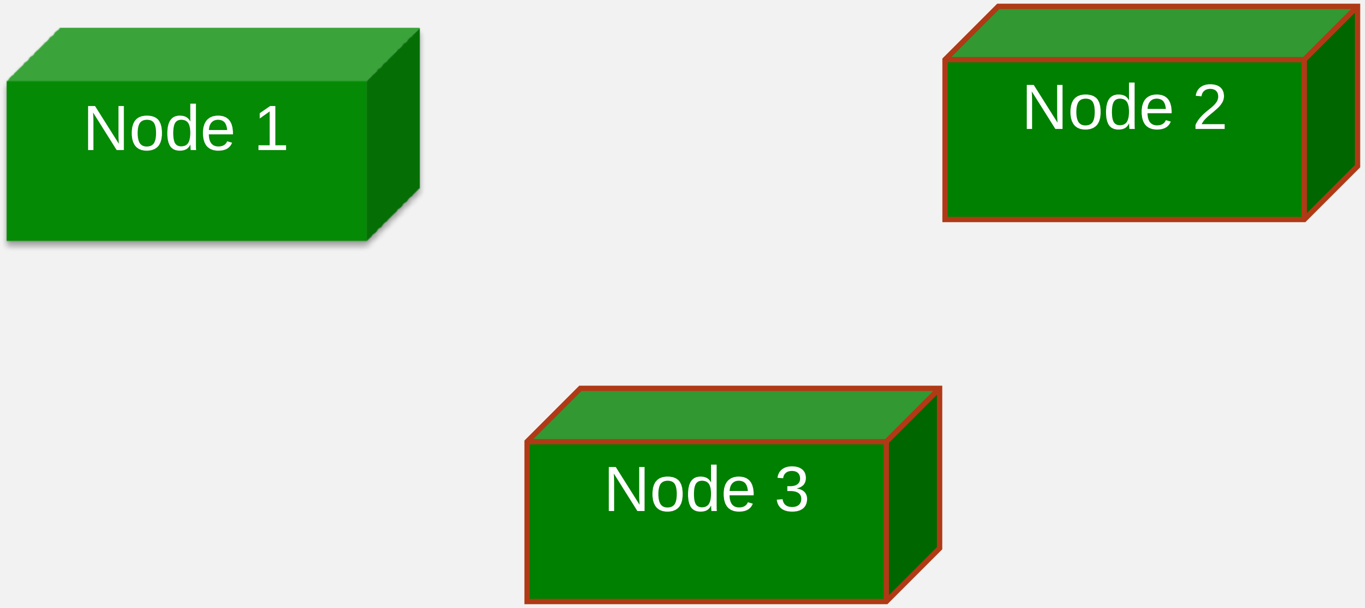
- Why?
- What is it?
- Configuration Options

mongoDB

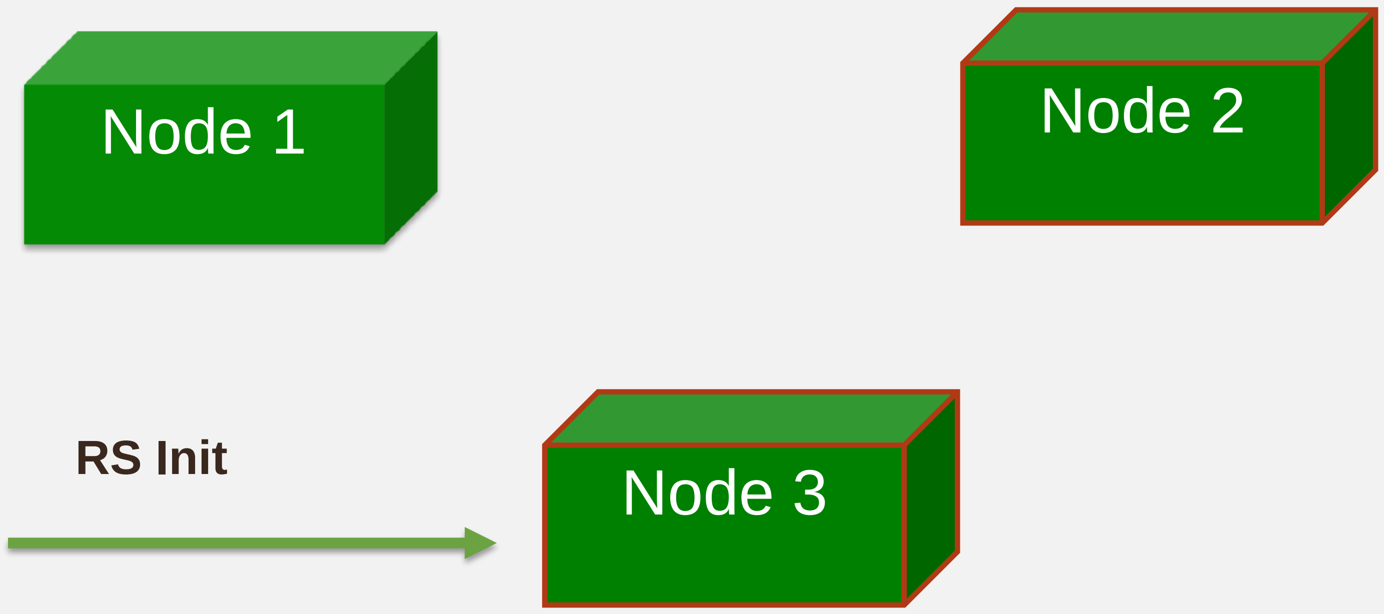
Why Replication?

- How many have faced node failures?
- How many have been woken to do fail overs?
- How many have experienced issues due to n/w latency?
- Different uses for data
  - Normal processing
  - Simple analytics

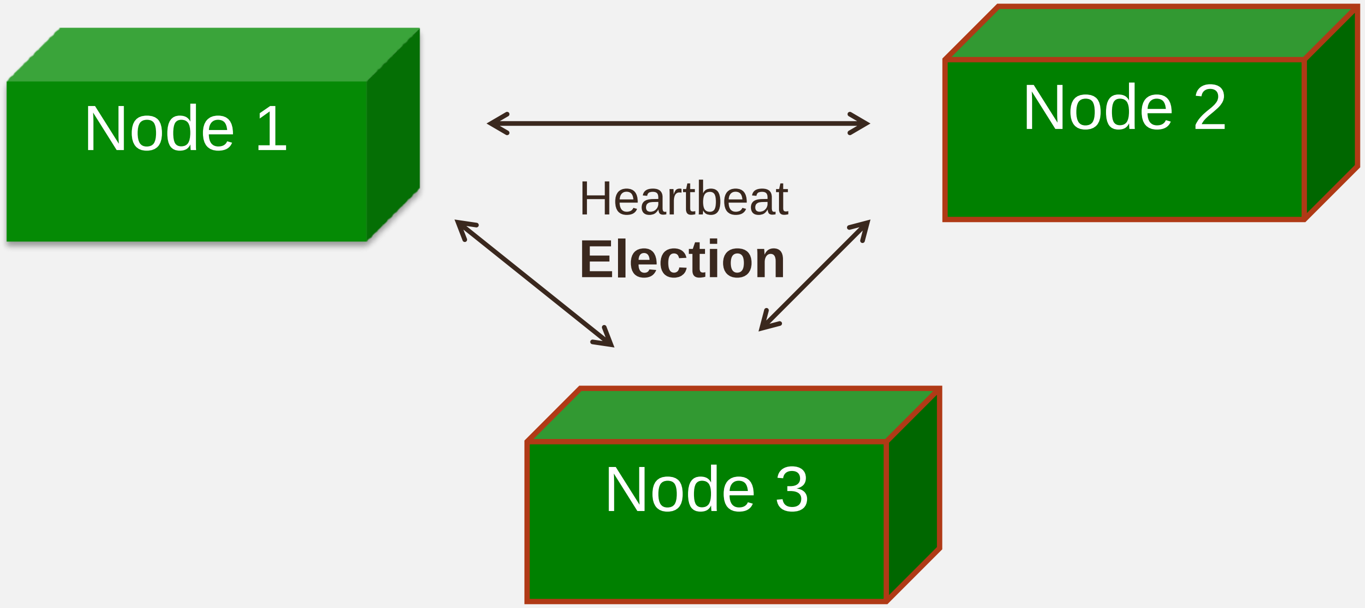
Replica Set - Creation



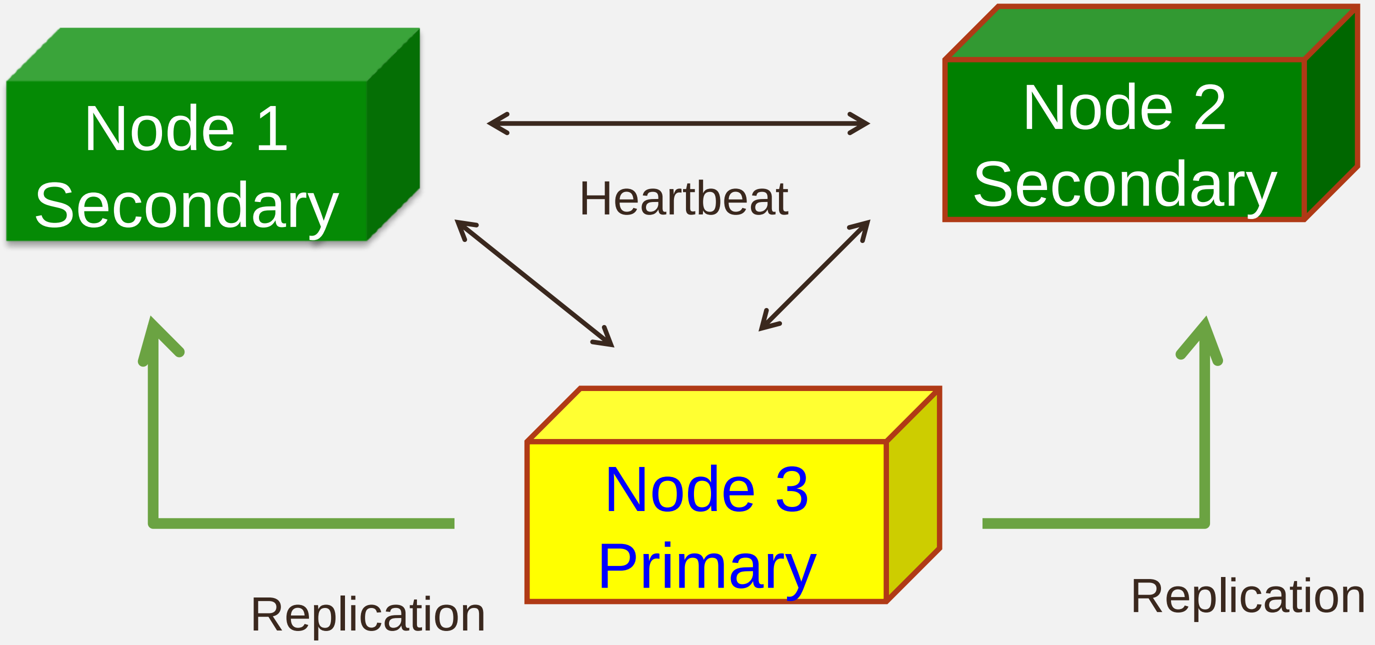
Replica Set - Initialize



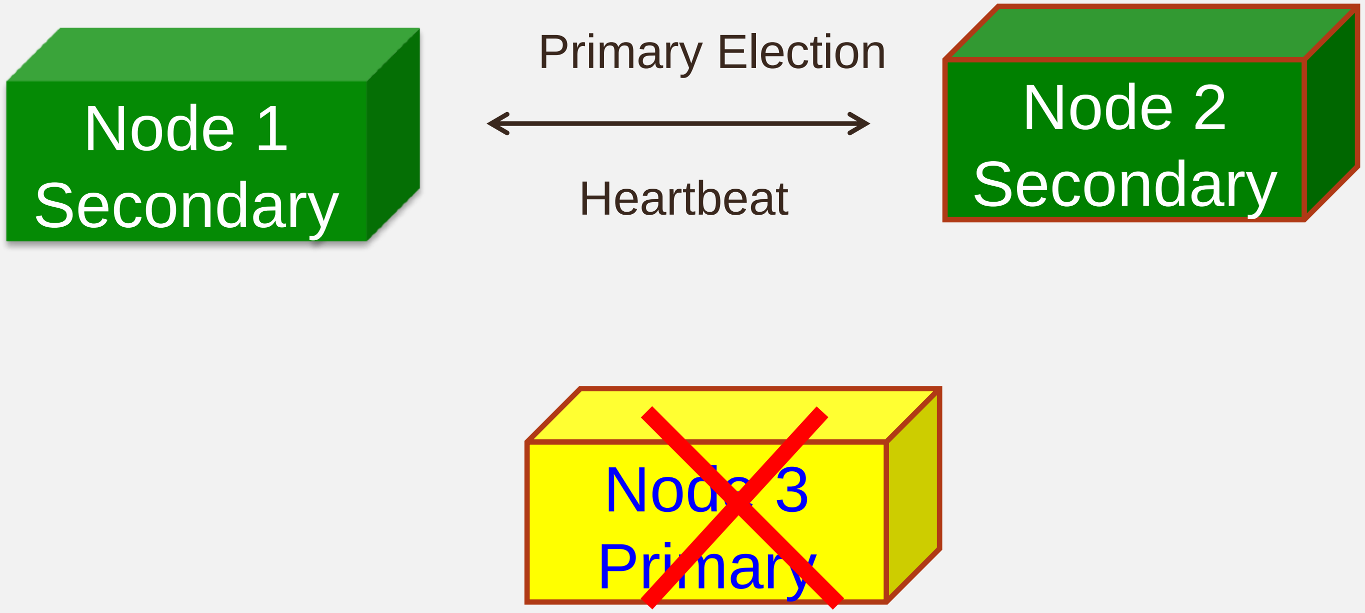
Replica Set - Initializing



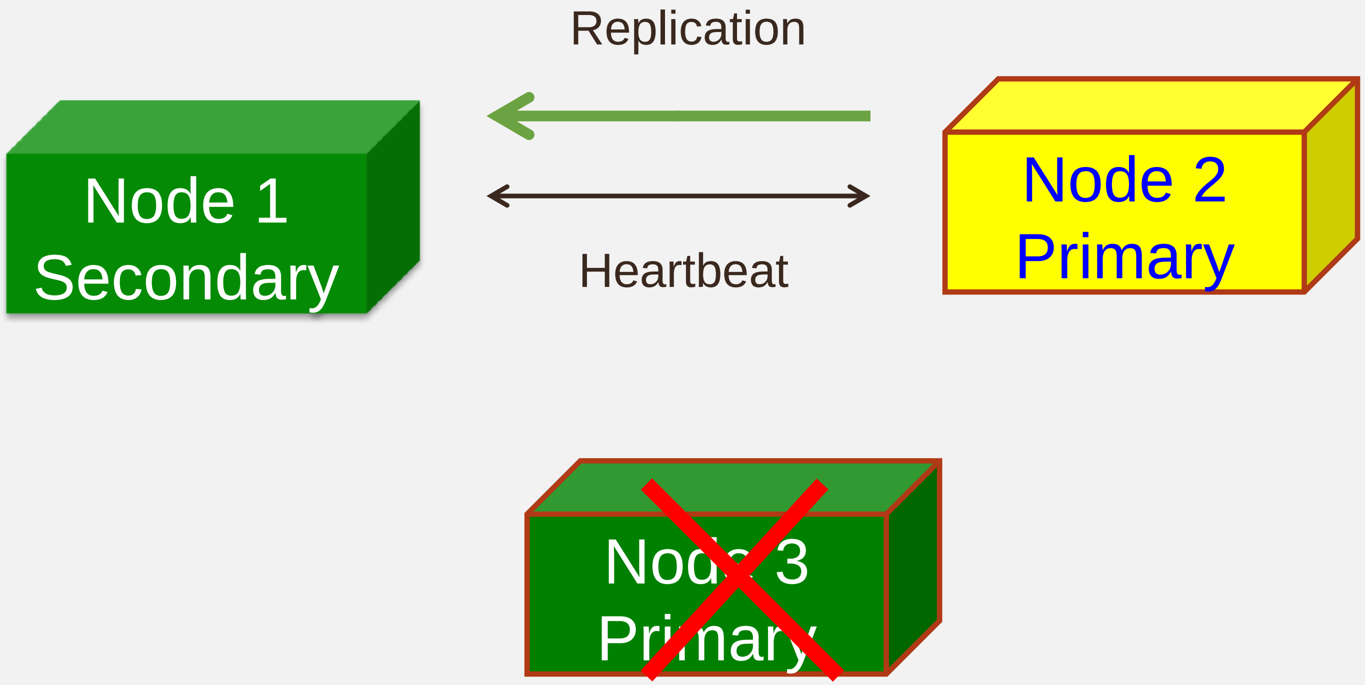
Replica Set - Initialized



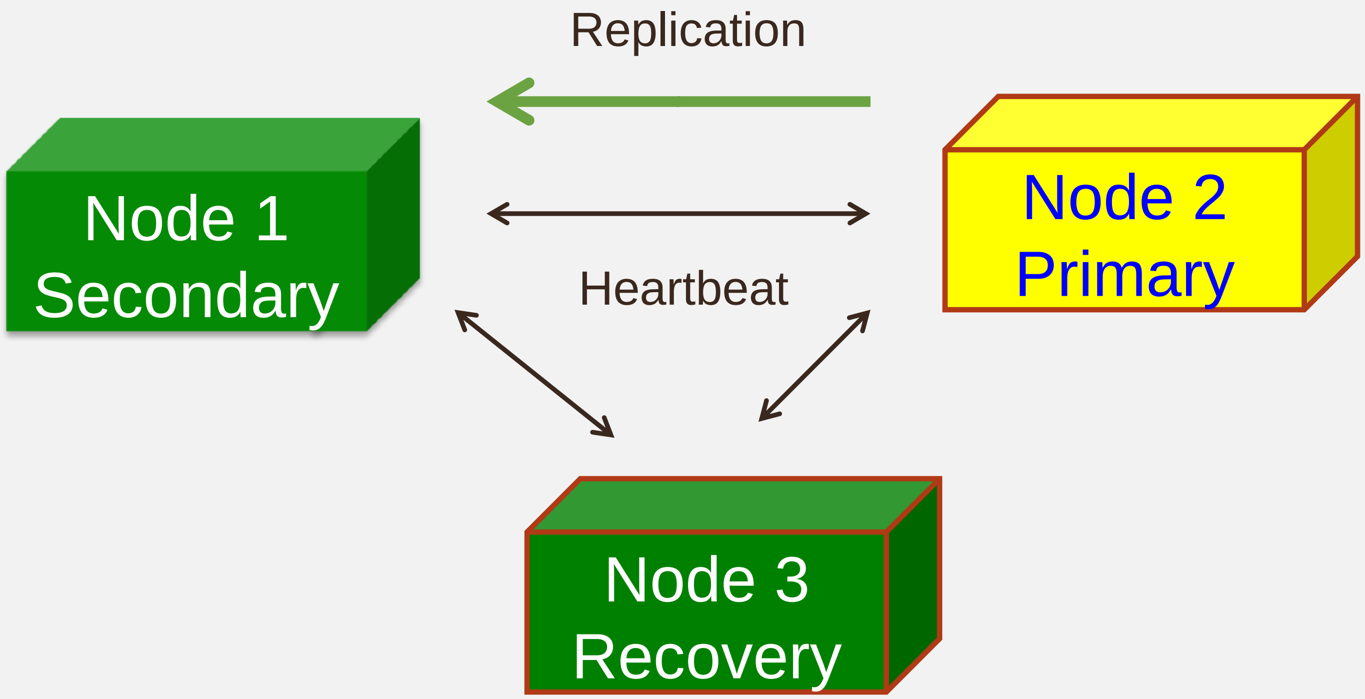
Replica Set - Failure



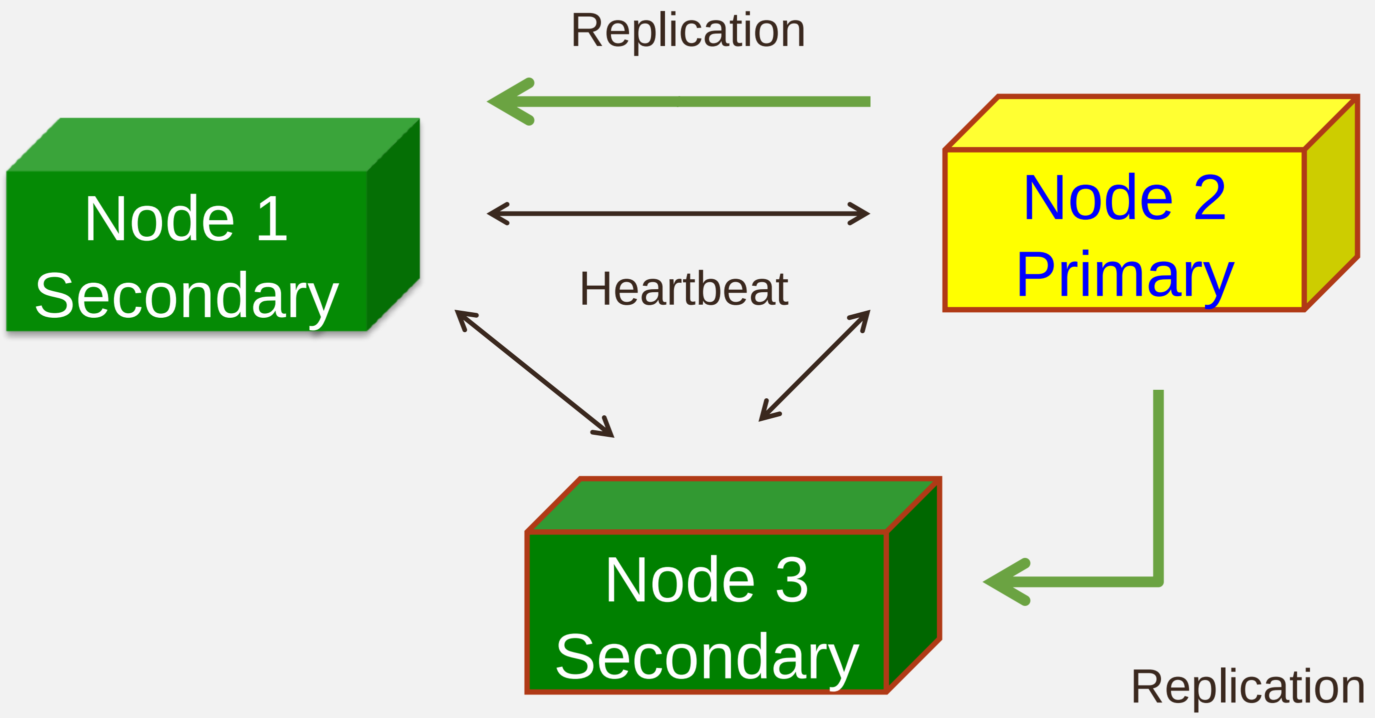
Replica Set - Failover



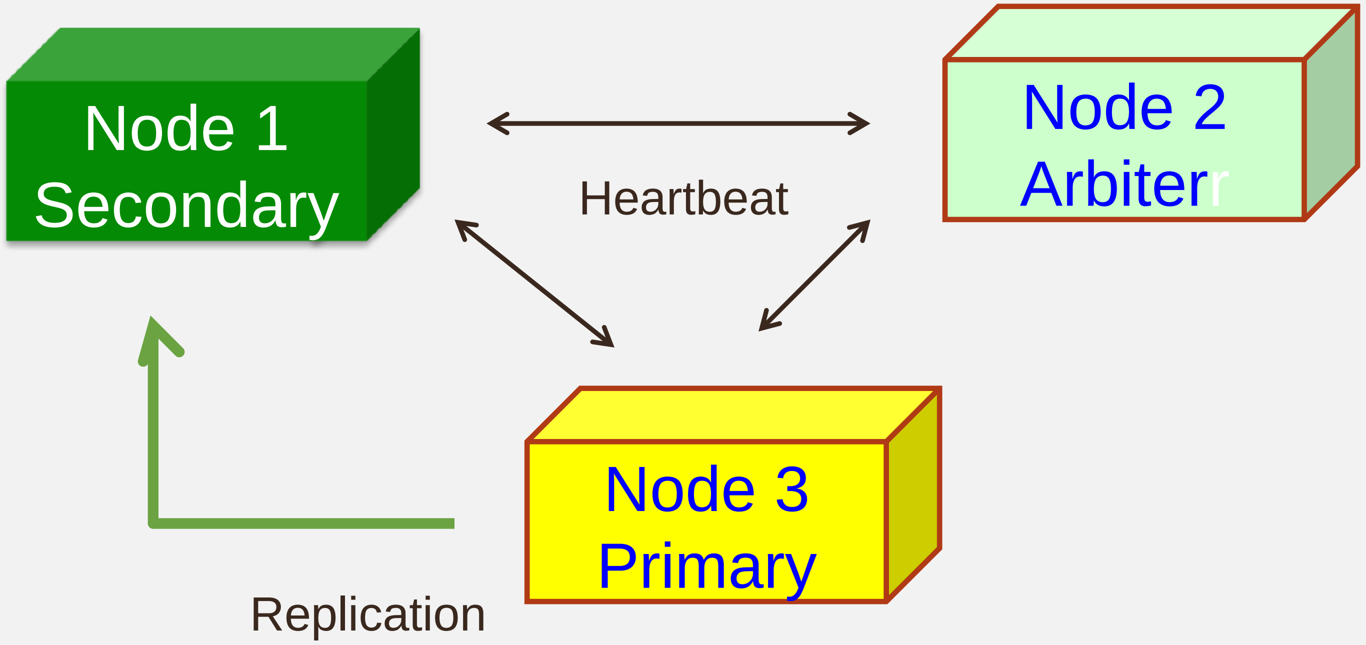
Replica Set - Recovery



Replica Set – Recovery Complete



Replica Set – Member Roles



Replica Set – 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)
```

Replica Set – 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

Replica Set – 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

Replica Set – 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

Replica Set – Configuration Options

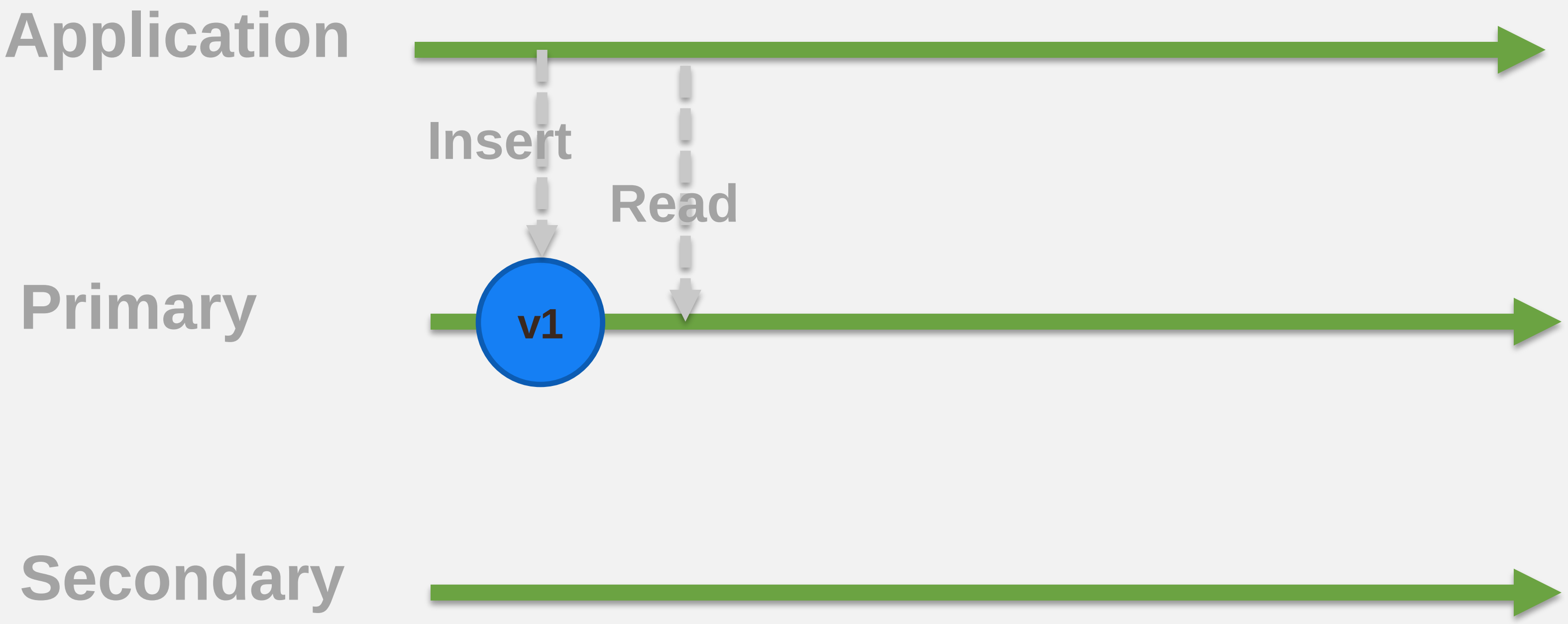
```
> conf = {
    _id : "mySet”,
    members : [
        {_id : 0, host : "A”, priority : 3},
        {_id : 1, host : "B", priority : 2},
        {_id : 2, host : "C”},
        {_id : 3, host : "D", hidden : true},
        {_id : 4, host : "E", hidden : true, slaveDelay : 3600}
     ]
}

> rs.initiate(conf)
```

Backup node

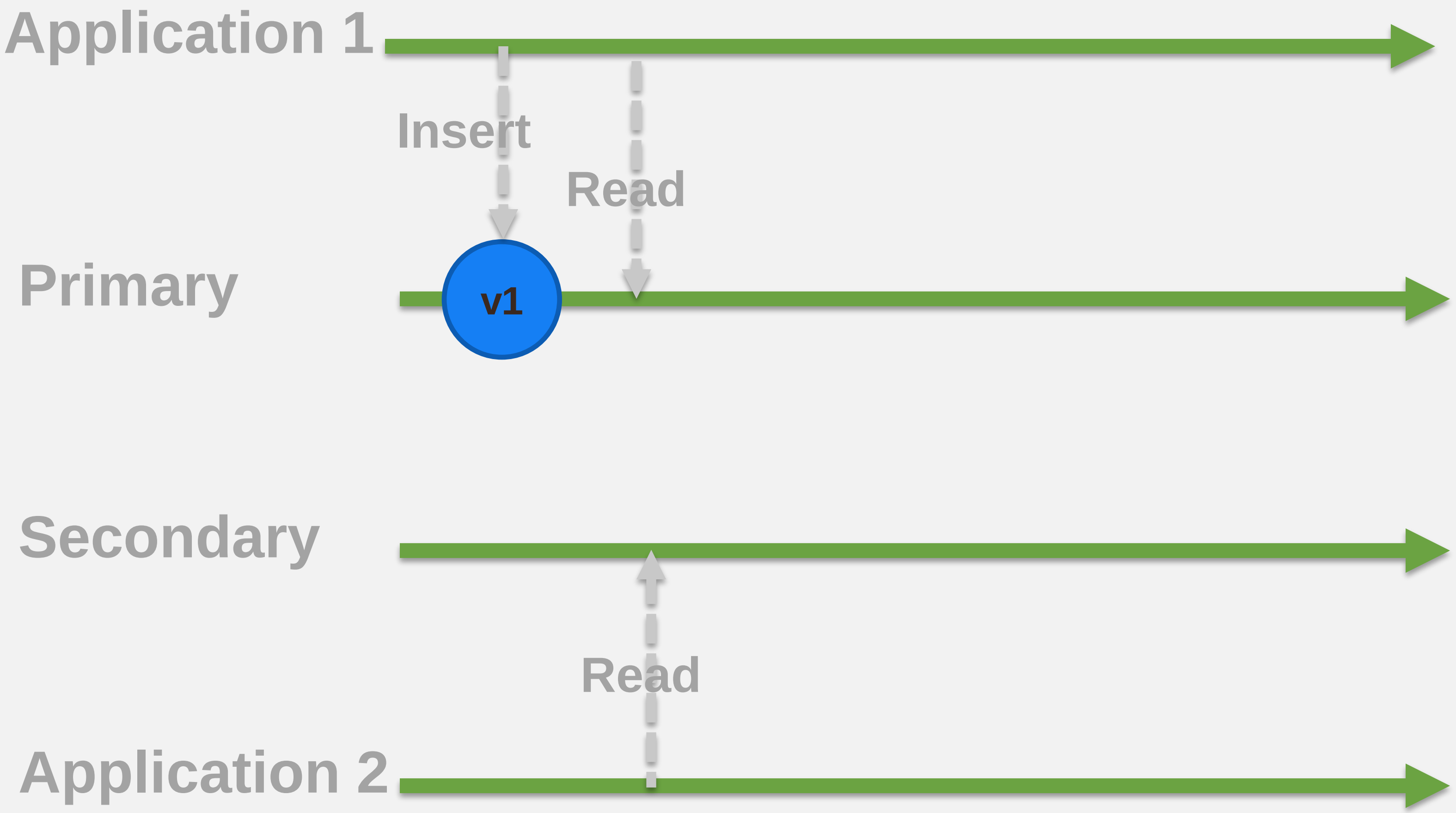
Developing with Replica Sets

- Consistency
- Write Preference
- Read Preference

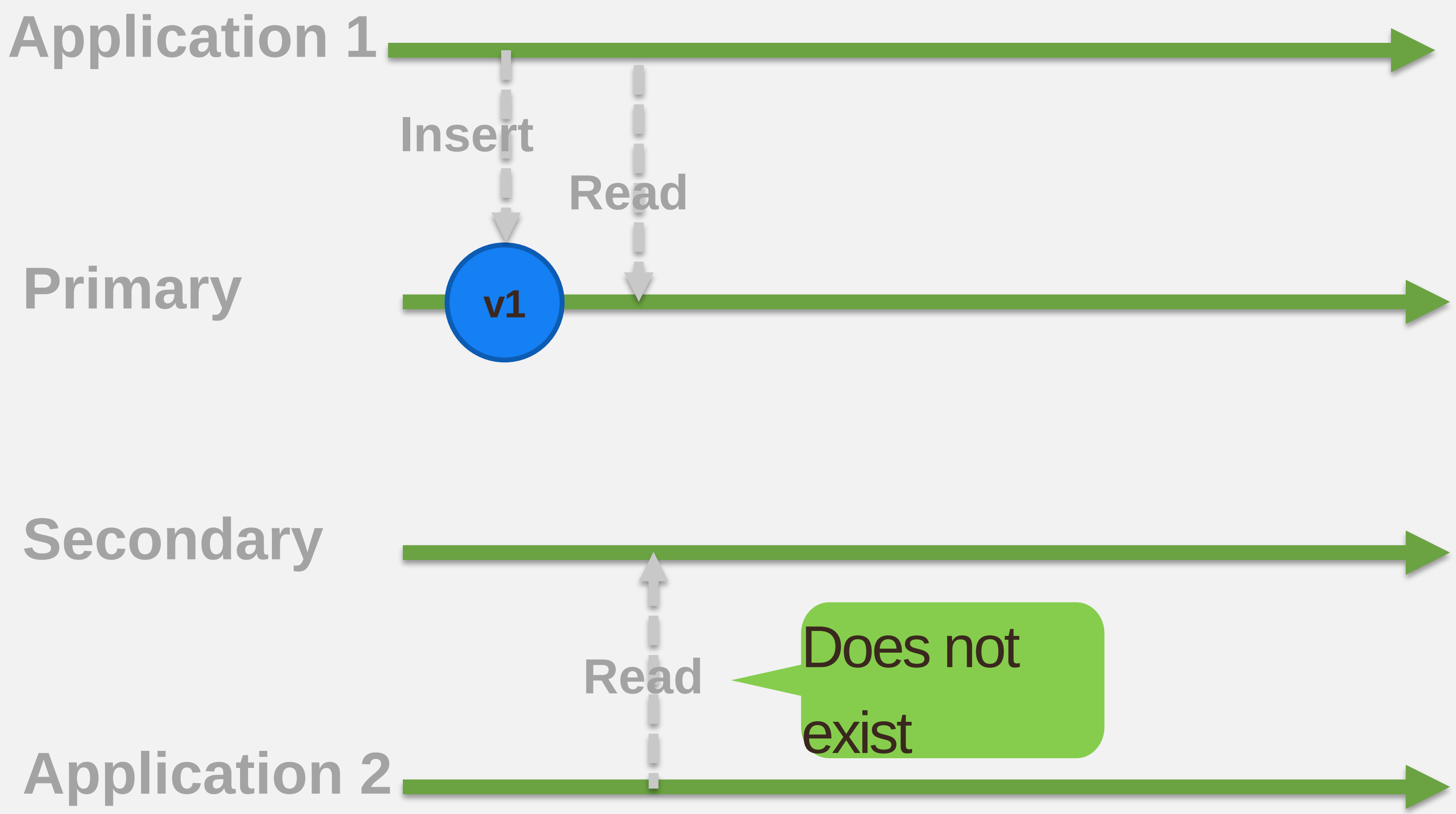
Consistency – Strong



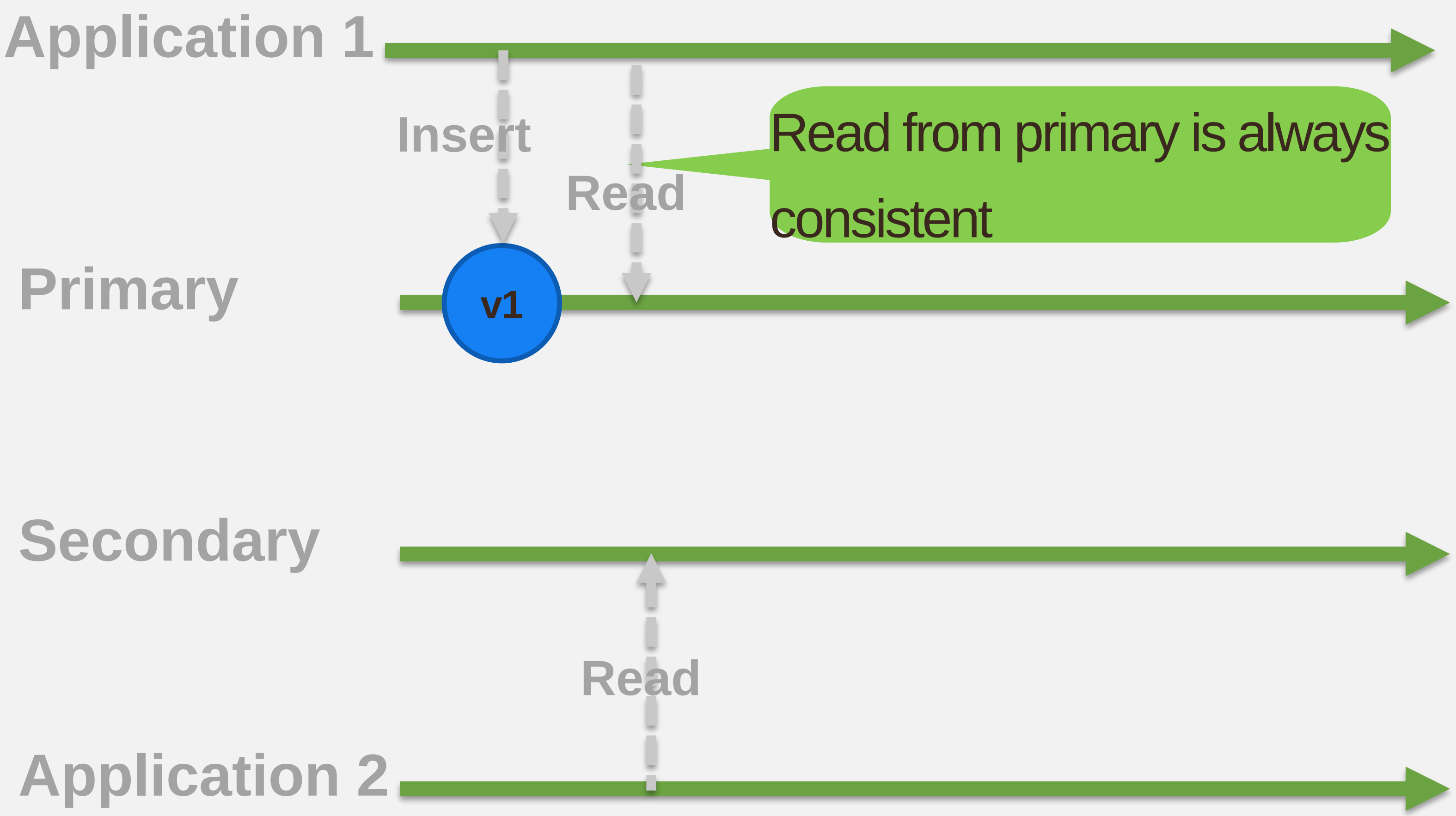
Consistency – Delayed



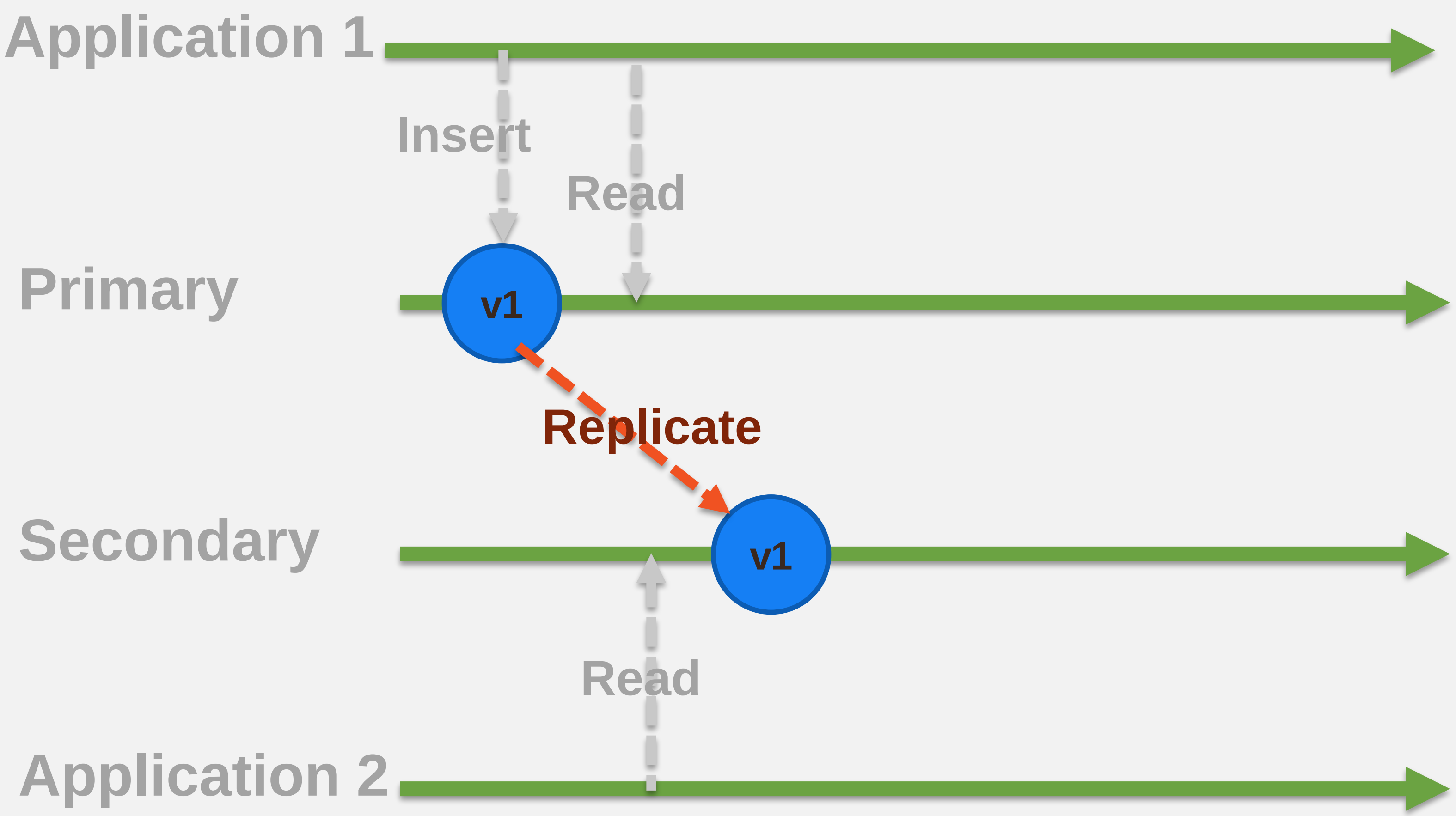
Consistency – Delayed



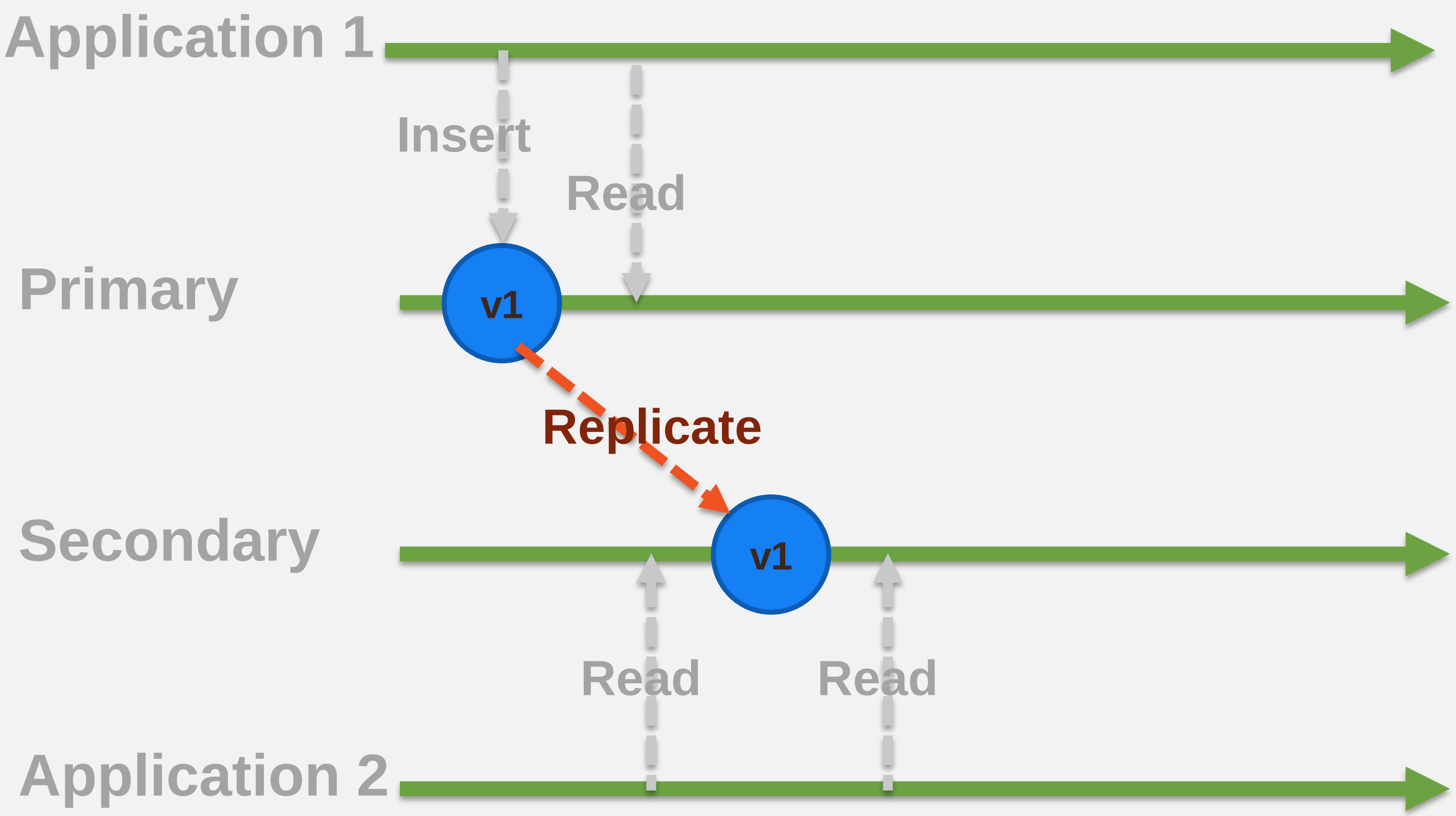
Consistency – Delayed



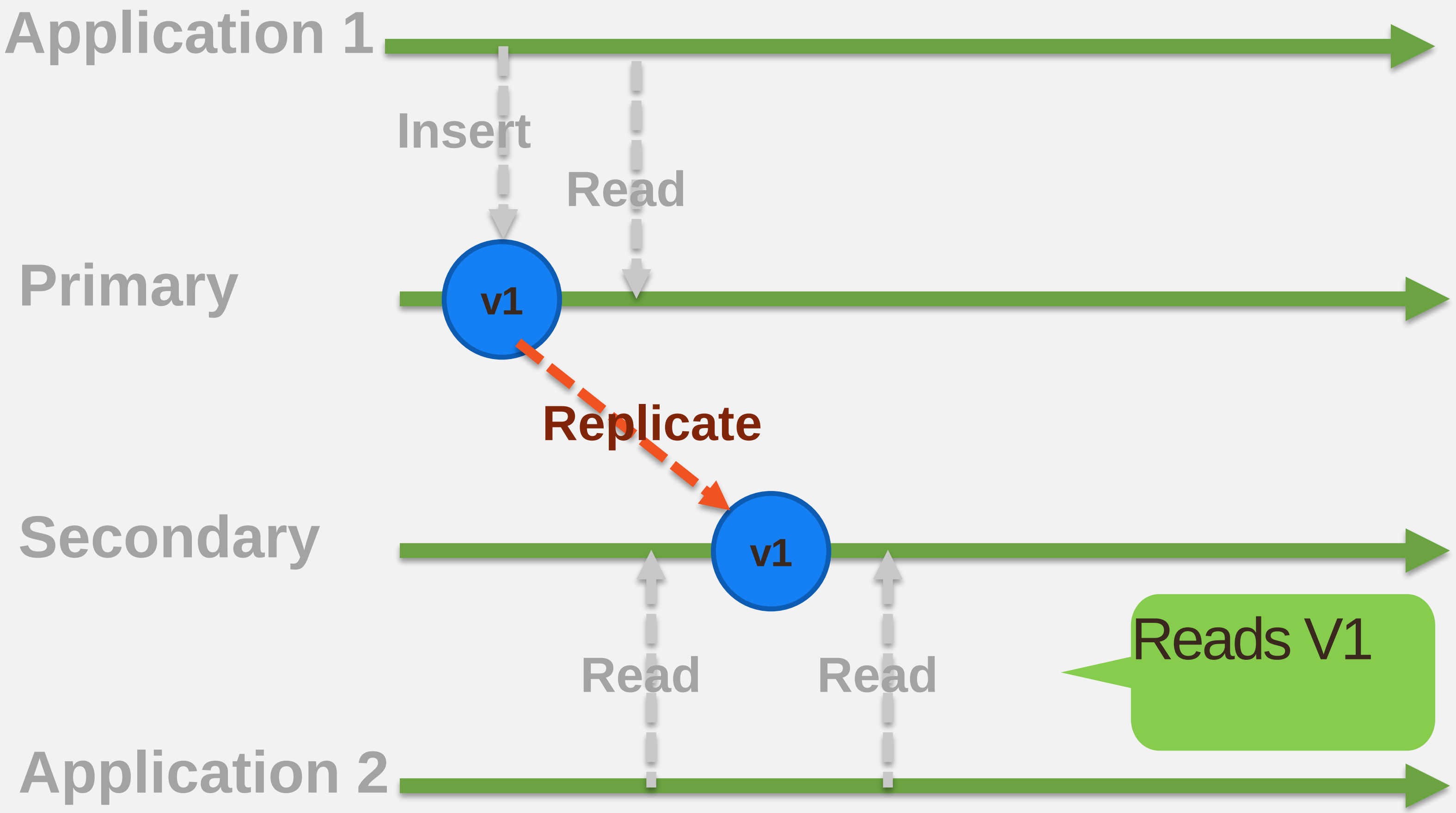
Consistency – Delayed



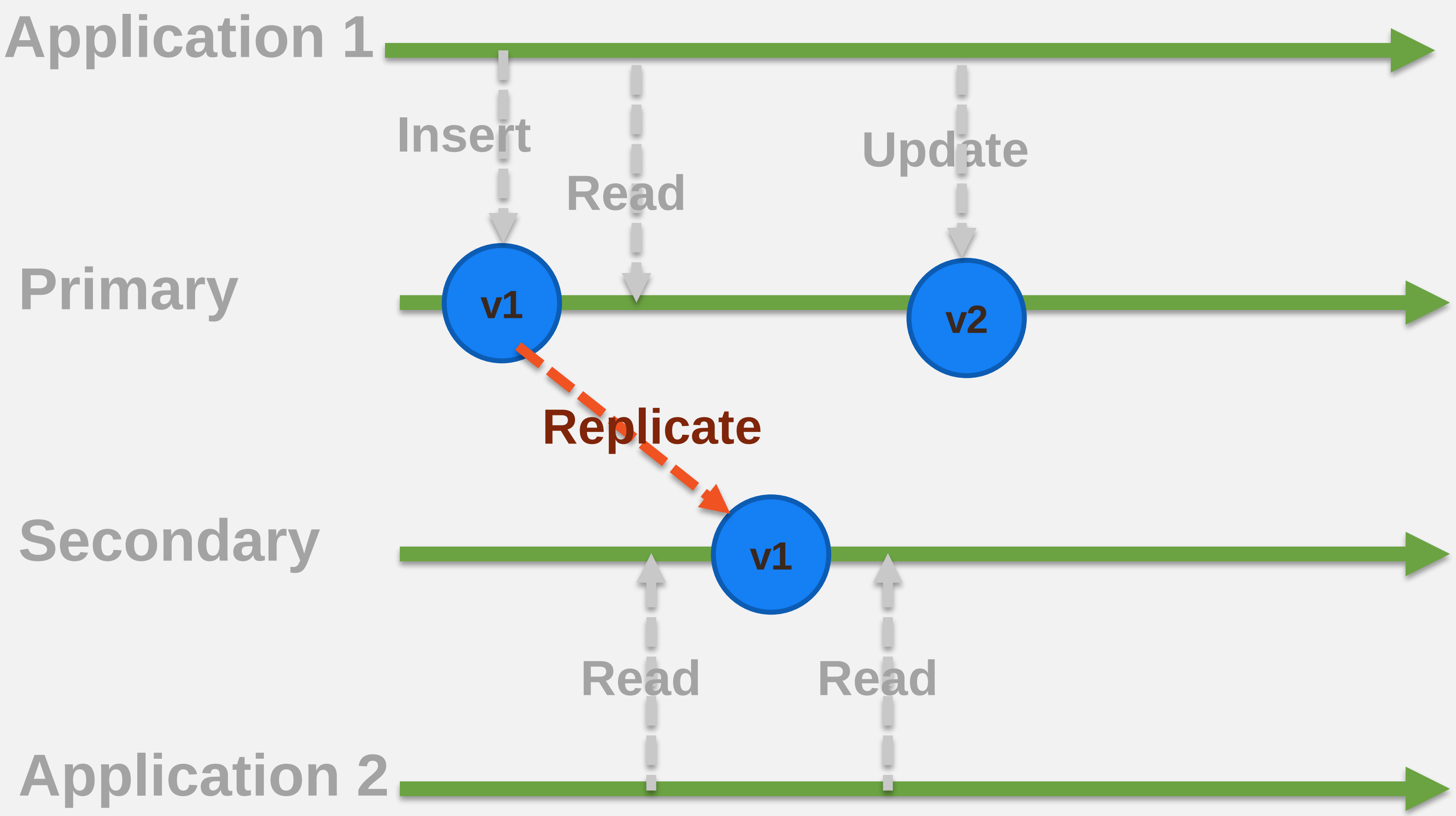
Consistency – Delayed



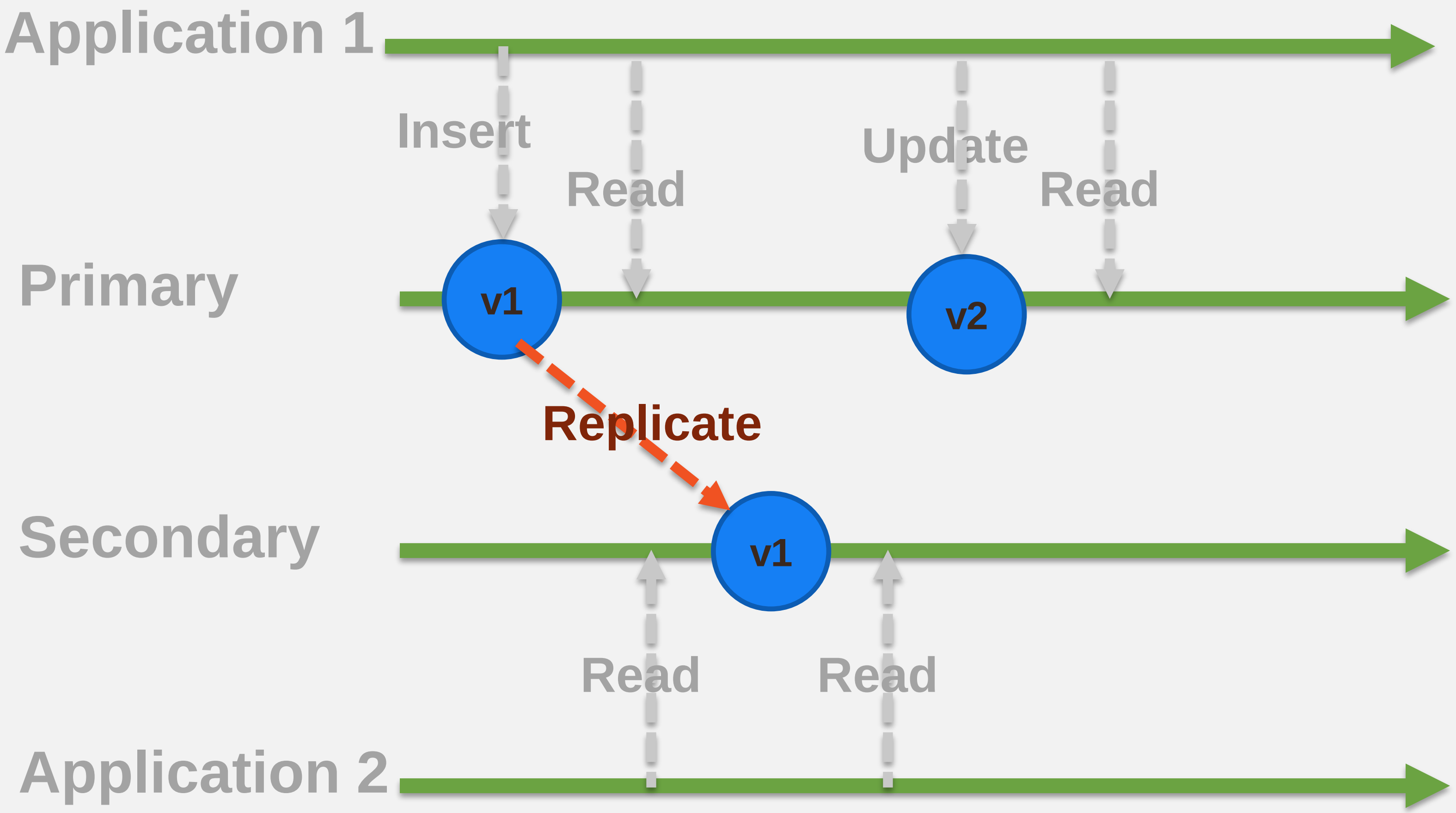
Consistency – Delayed



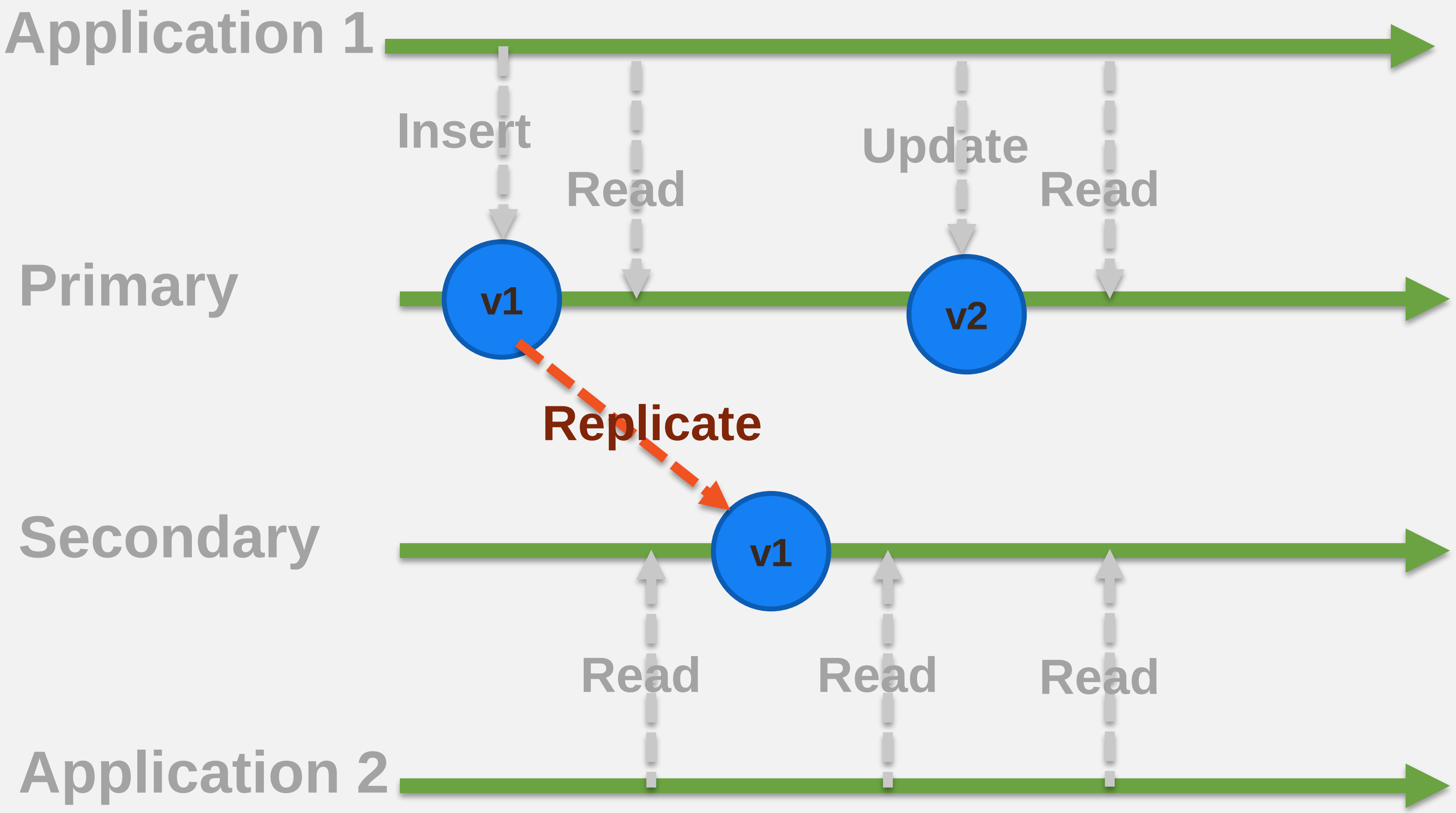
Consistency – Delayed



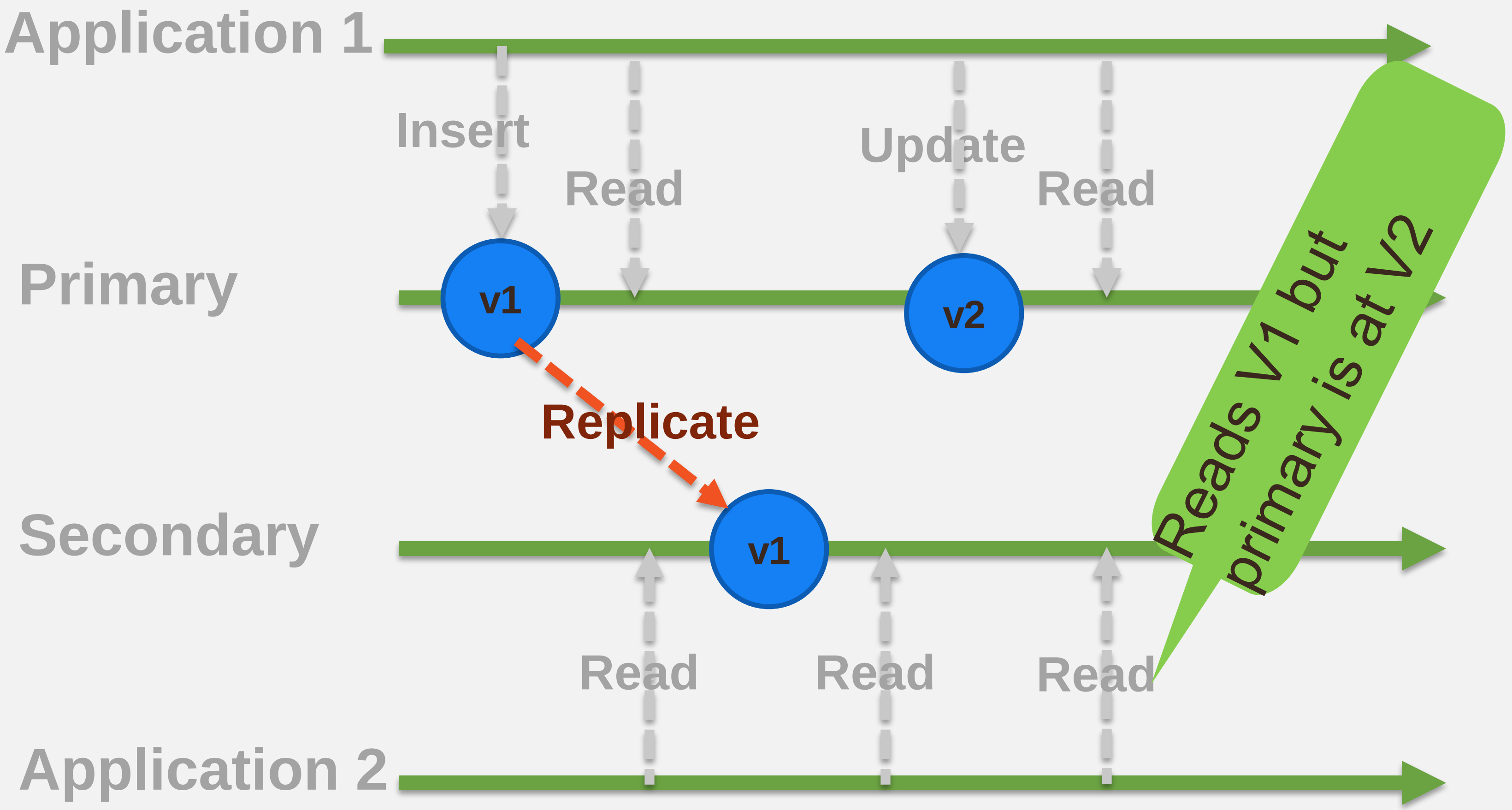
Consistency – Delayed



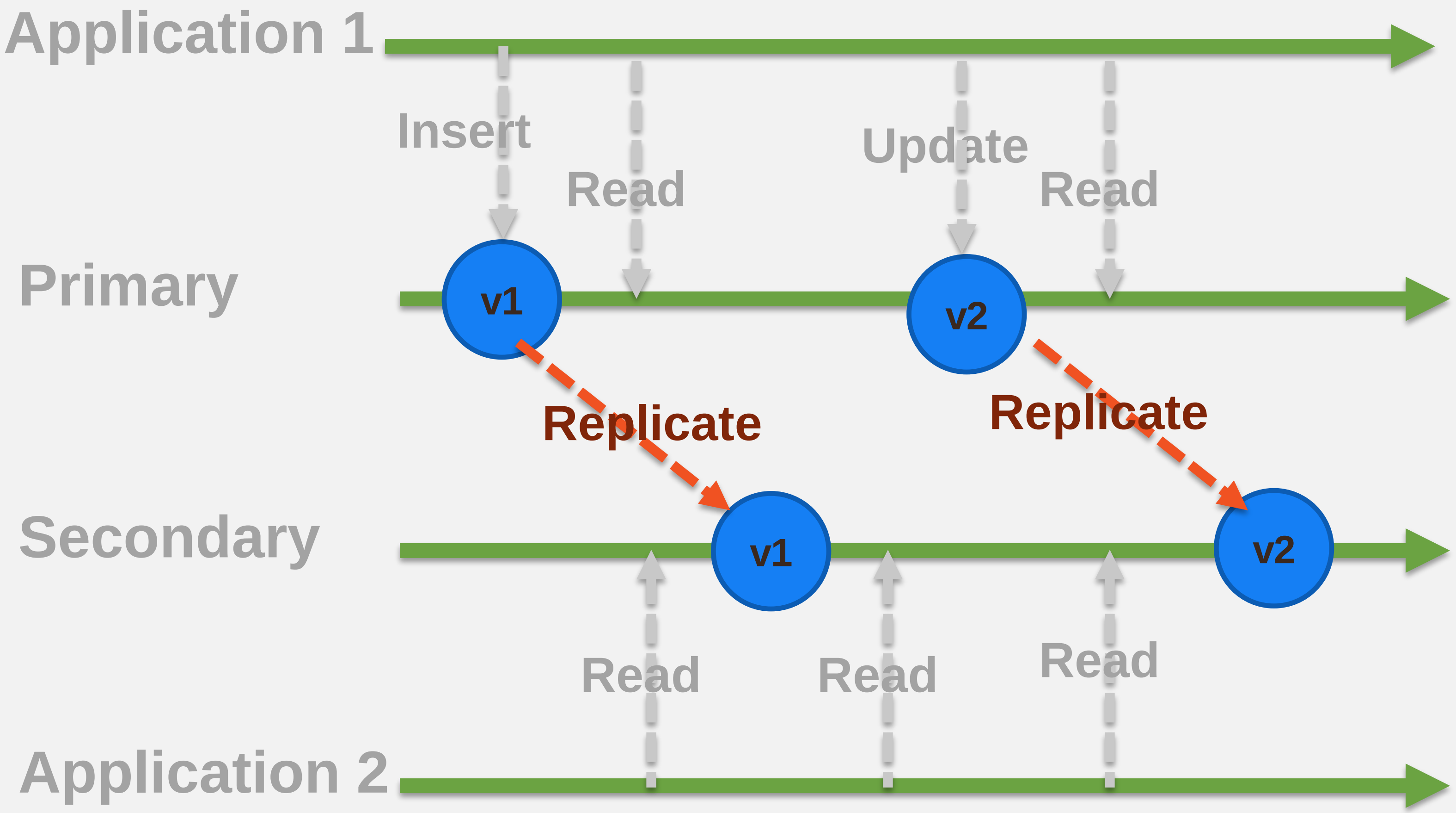
Consistency – Delayed



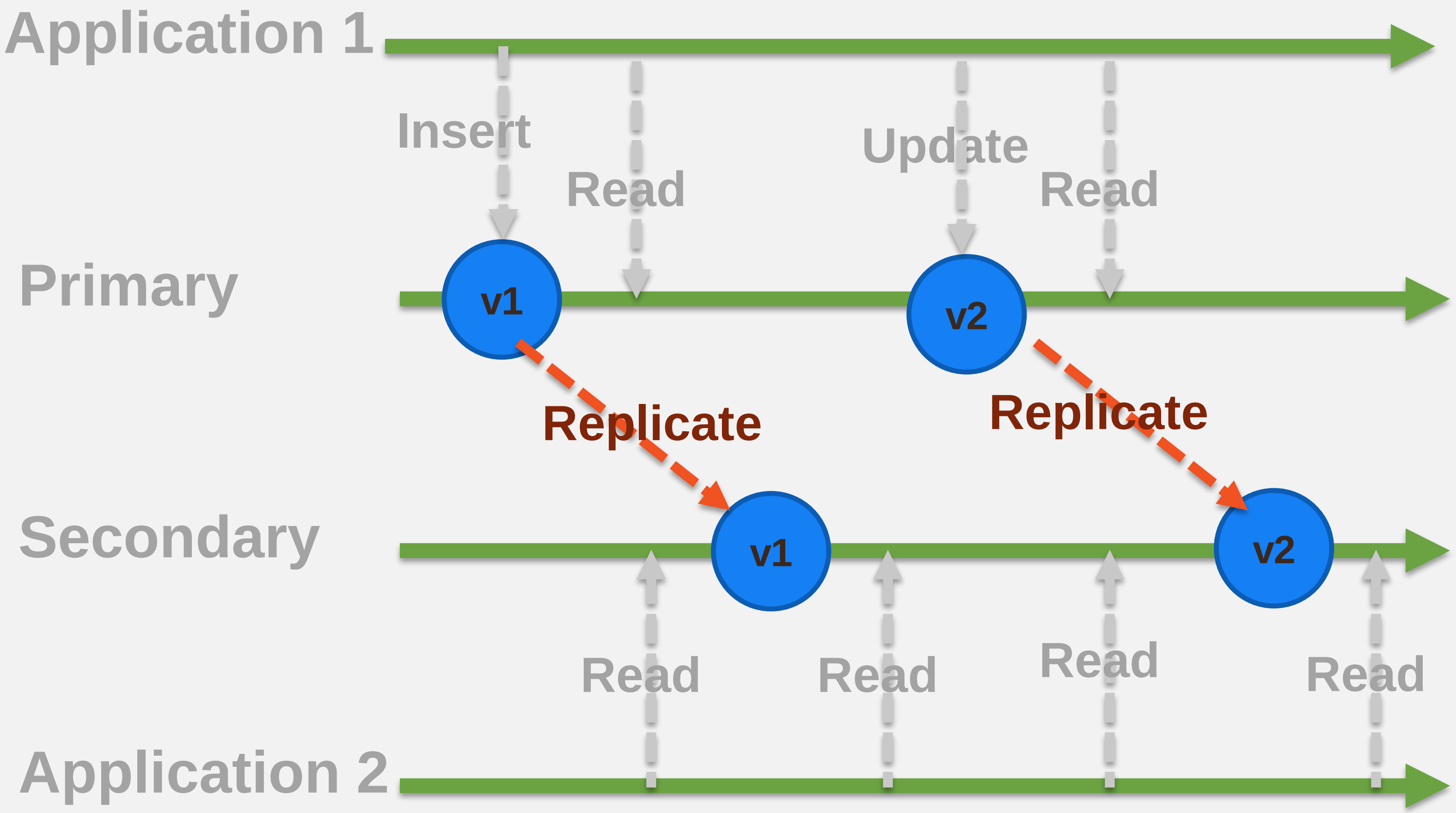
Consistency – Delayed



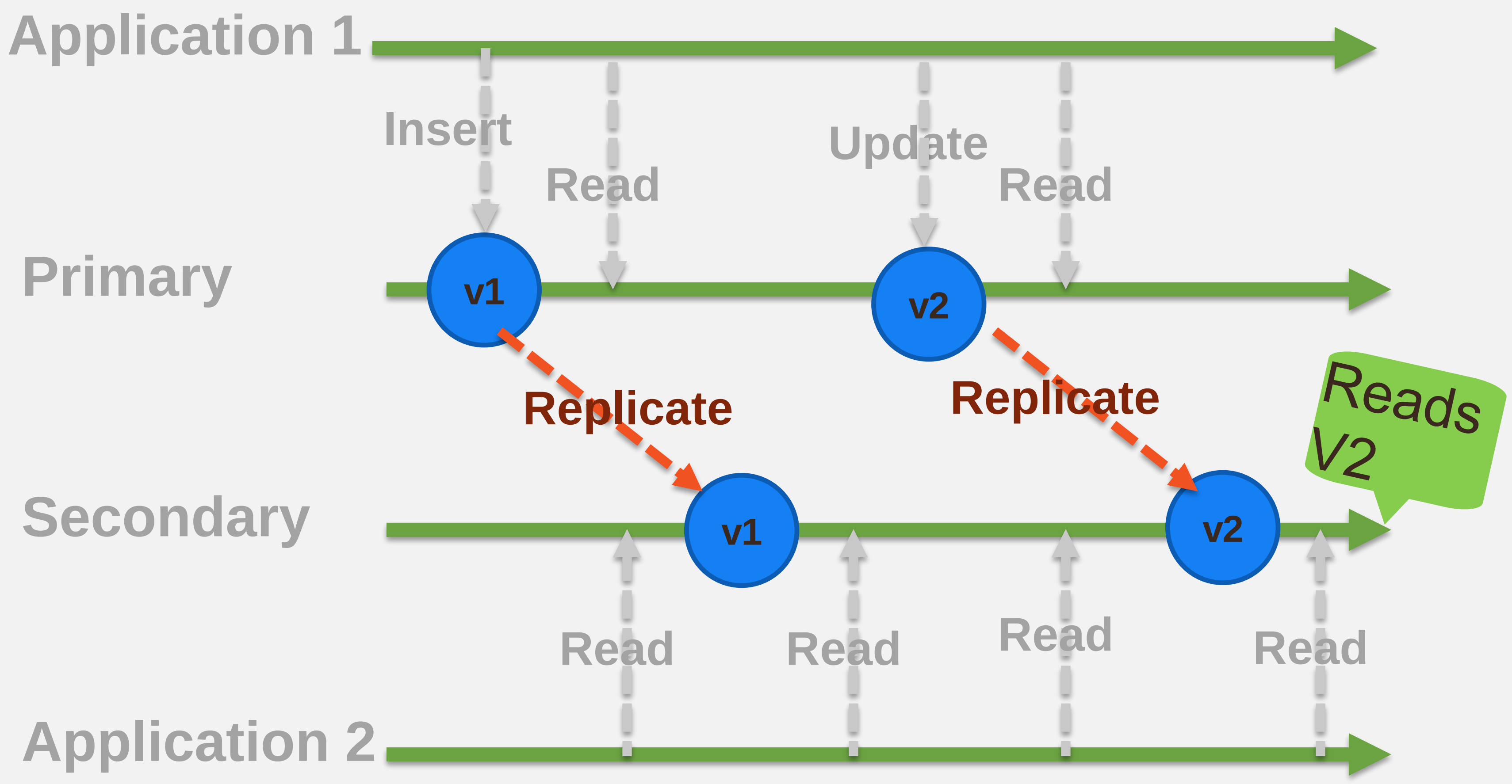
Consistency – Delayed



Consistency – Delayed



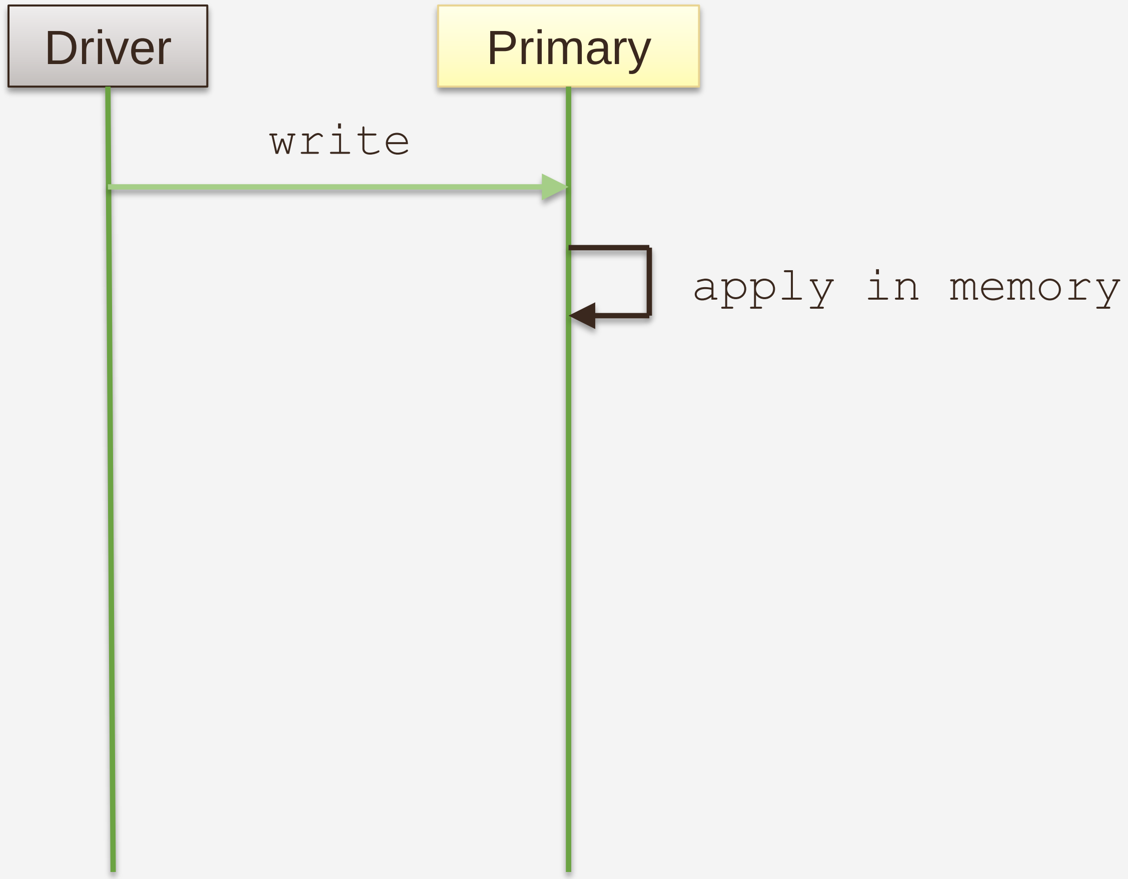
Consistency – Delayed



Write Preference

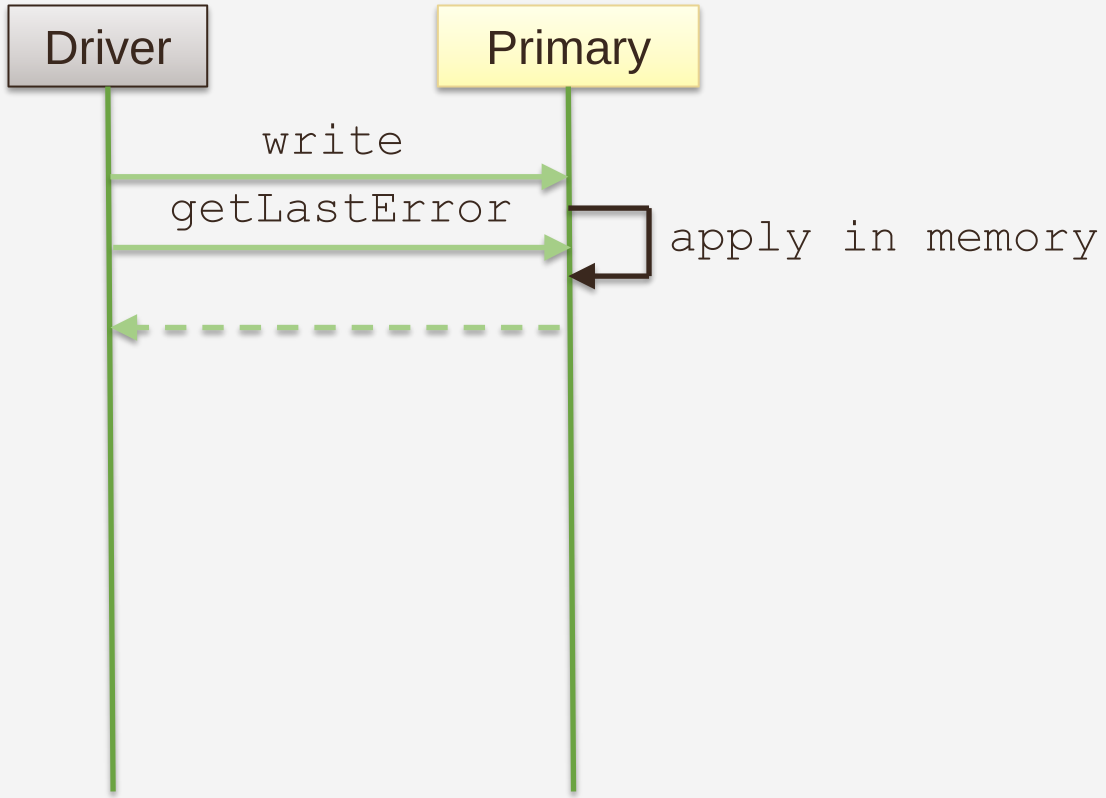
- Network acknowledged
- Wait for error
- Wait for journal sync
- Wait for replication

Network Acknowledged

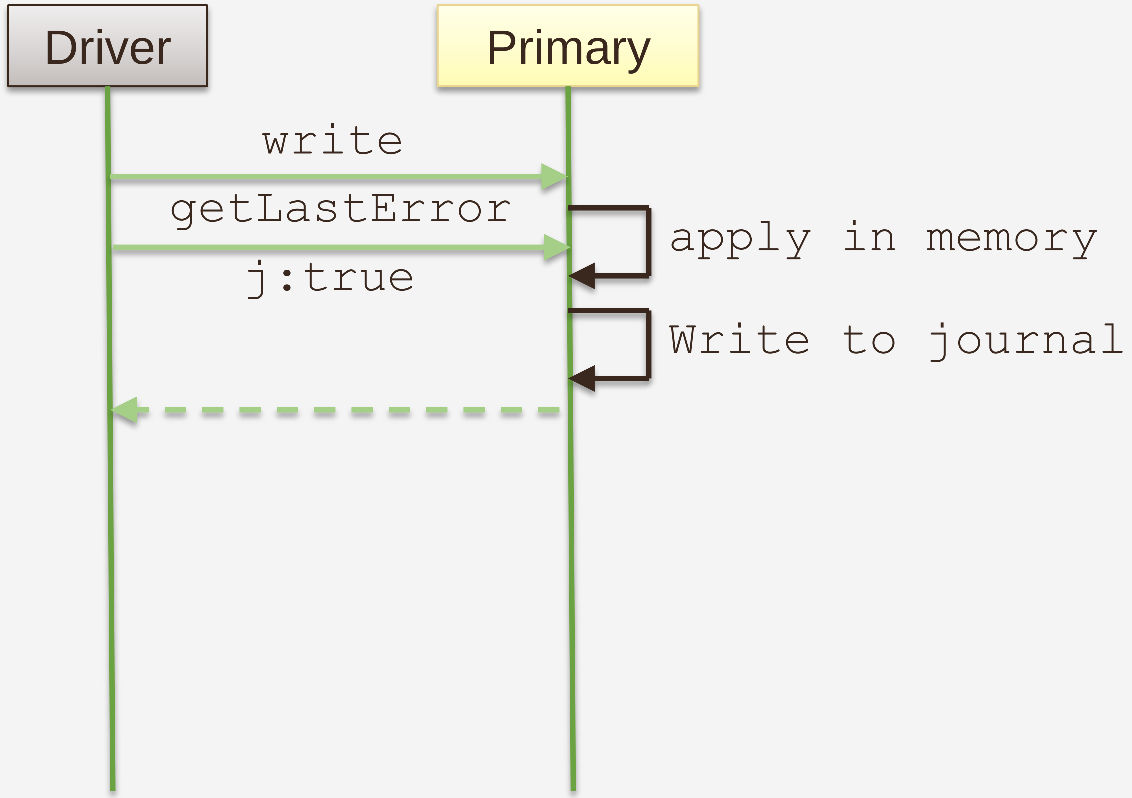


mongoDB

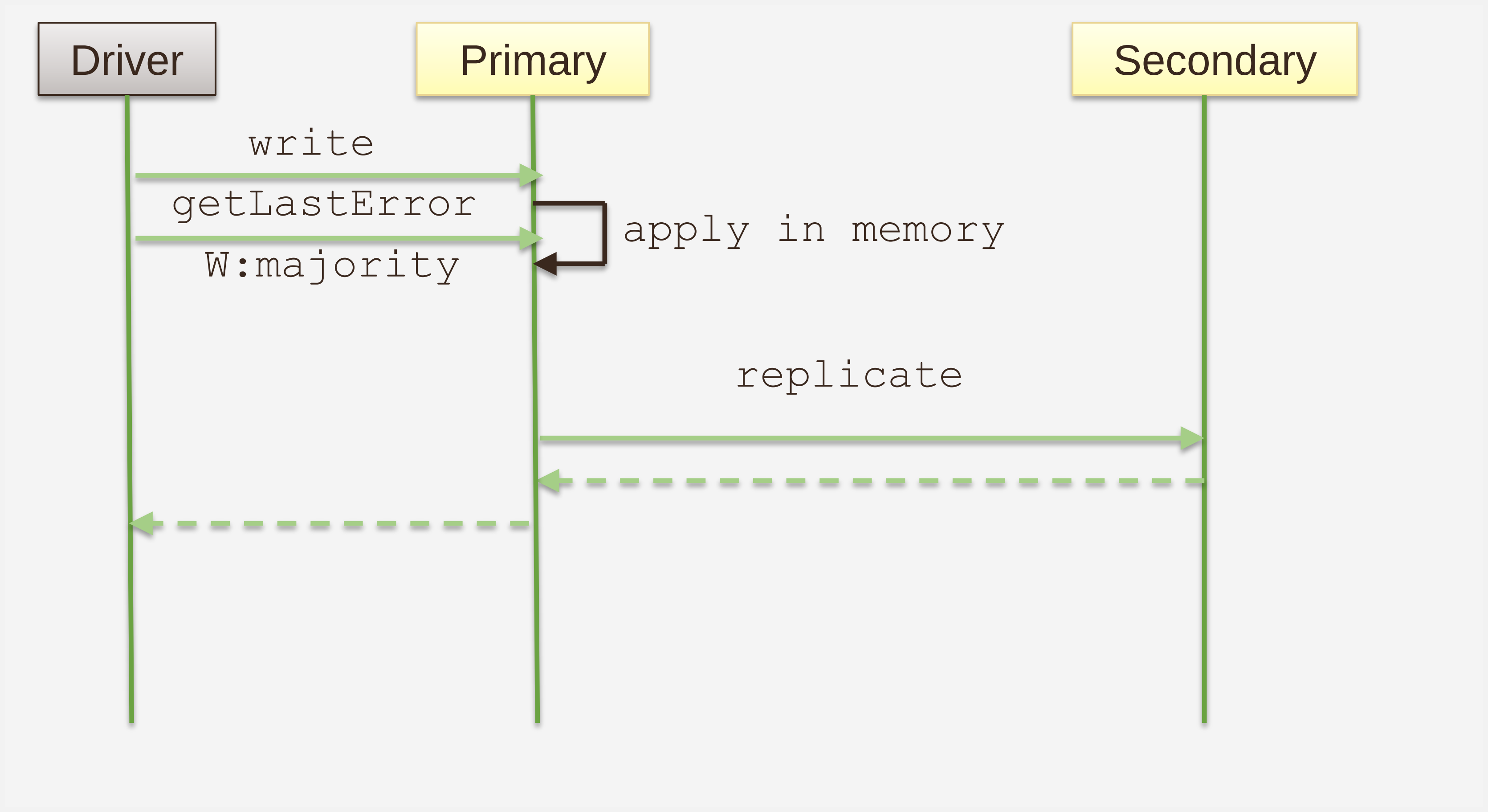
Wait for error



Wait for journal sync



Wait for replication



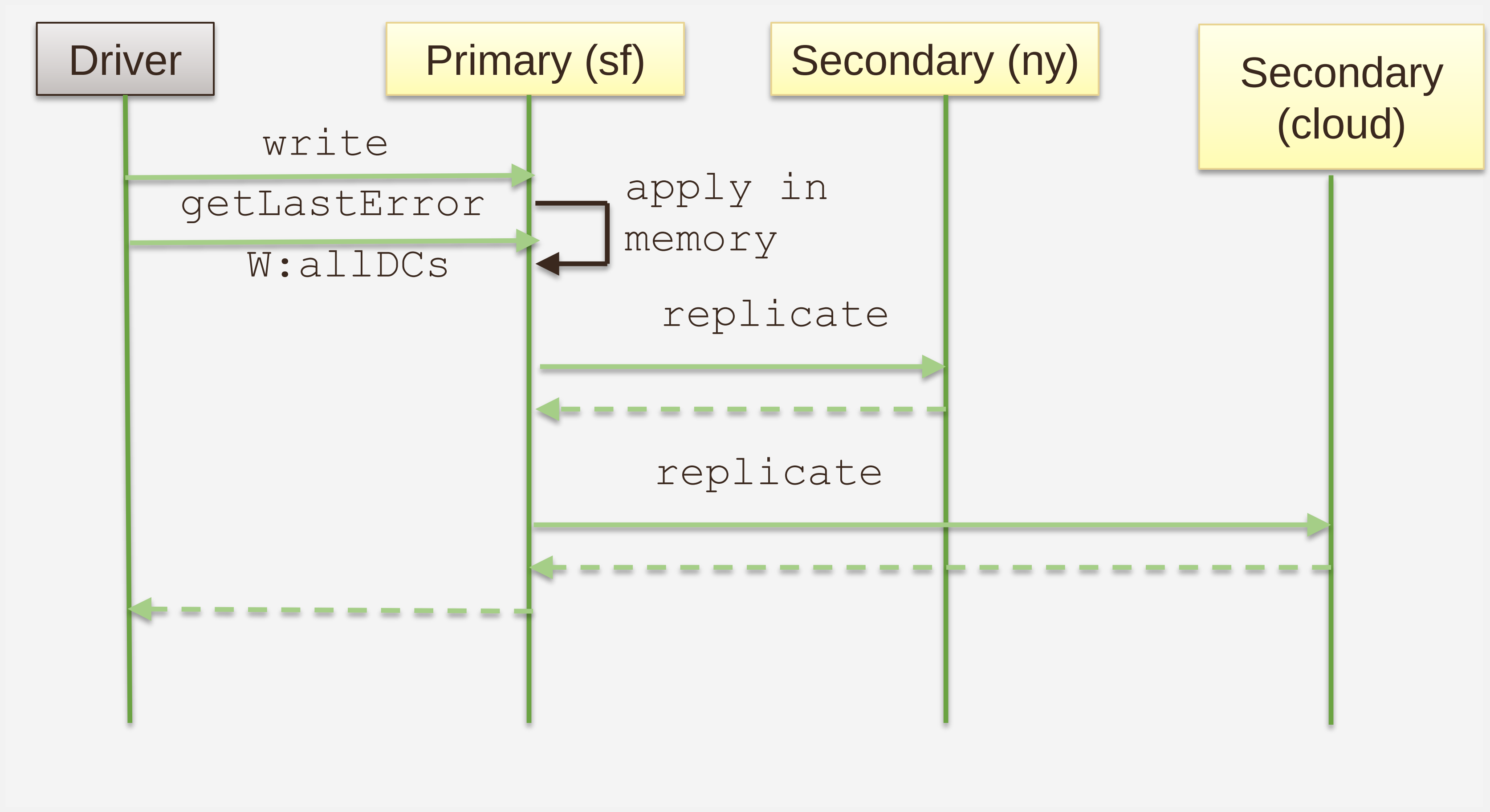
Tagging

- Since 2.0.0
- Control over where data is written to
- Each member can have one or more tags e.g.
  - Dc : "ny”
  - dc: "ny", ip: "192.168", rack: "row3rk7”
- Replica set defines rules for where data resides
- 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 : "allDCs"})
```

Wait for replication with tags



mongoDB

Read Preference

- 5 modes (new in 2.2)
  - PRIMARY(only) - Default
  - PRIMARYPREFERRED
  - SECONDARY (only)
  - SECONDARYPREFERRED
  - NEAREST

Tag sets

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

Operational Considerations

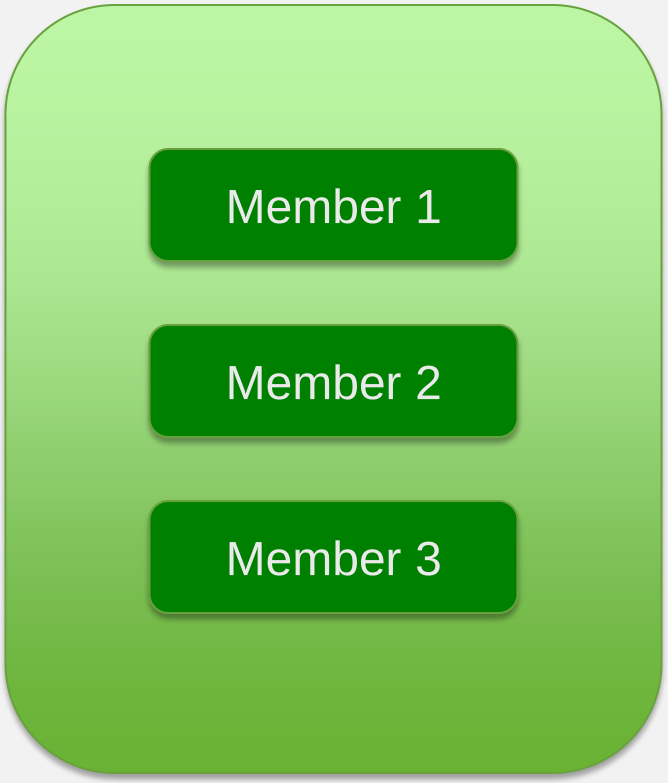
- Upgrade/Maintenance
- Common Deployment Scenarios

mongoDB

Maintenance and Upgrade

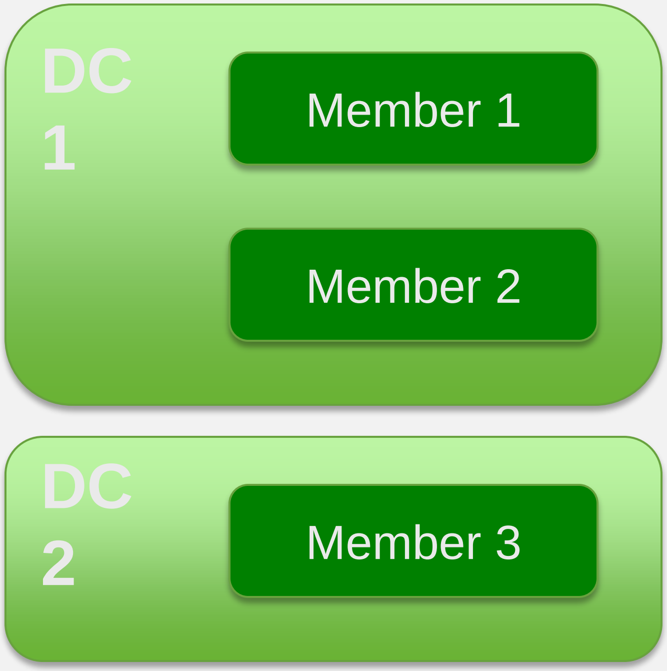
- No downtime
- Rolling upgrade/maintenance
  - Start with Secondary
  - Primary last

Replica Set – 1 Data Center



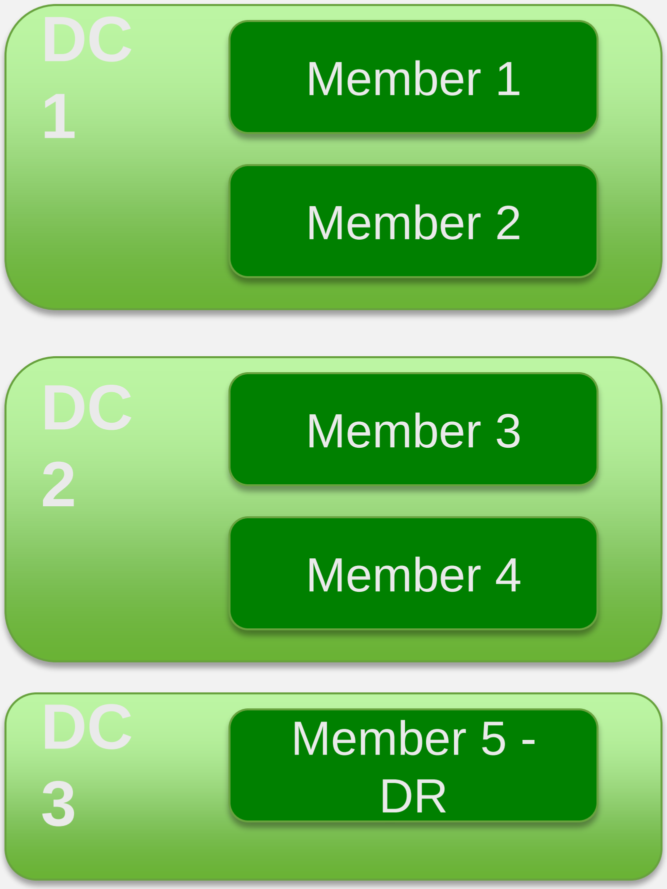
- Single datacenter
- Single switch & power
- Points of failure:
  - Power
  - Network
  - Datacenter
  - Two node failure
- Automatic recovery of single node crash

Replica Set – 2 data centers



- Multi datacenter
- 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

- Schema
- Oplog

mongoDB

Schema

- Local DB (not replicated)
  - system.replset
  - oplog.rs
    - Capped collection
    - Idempotent version of operation stored

Detections

- Heartbeat every 2 seconds
  - Times out in 10 seconds
- Missed heartbeat considered node down

Oplog

```
> 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 } } }
```

Oplog

```
> 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" } } }
```

#codemash

Coming Next:
Part 3.2 : Sharding

Sridhar Nanjundeswaran

Engineer, MongoDB Inc.

@snanjund

mongoDB