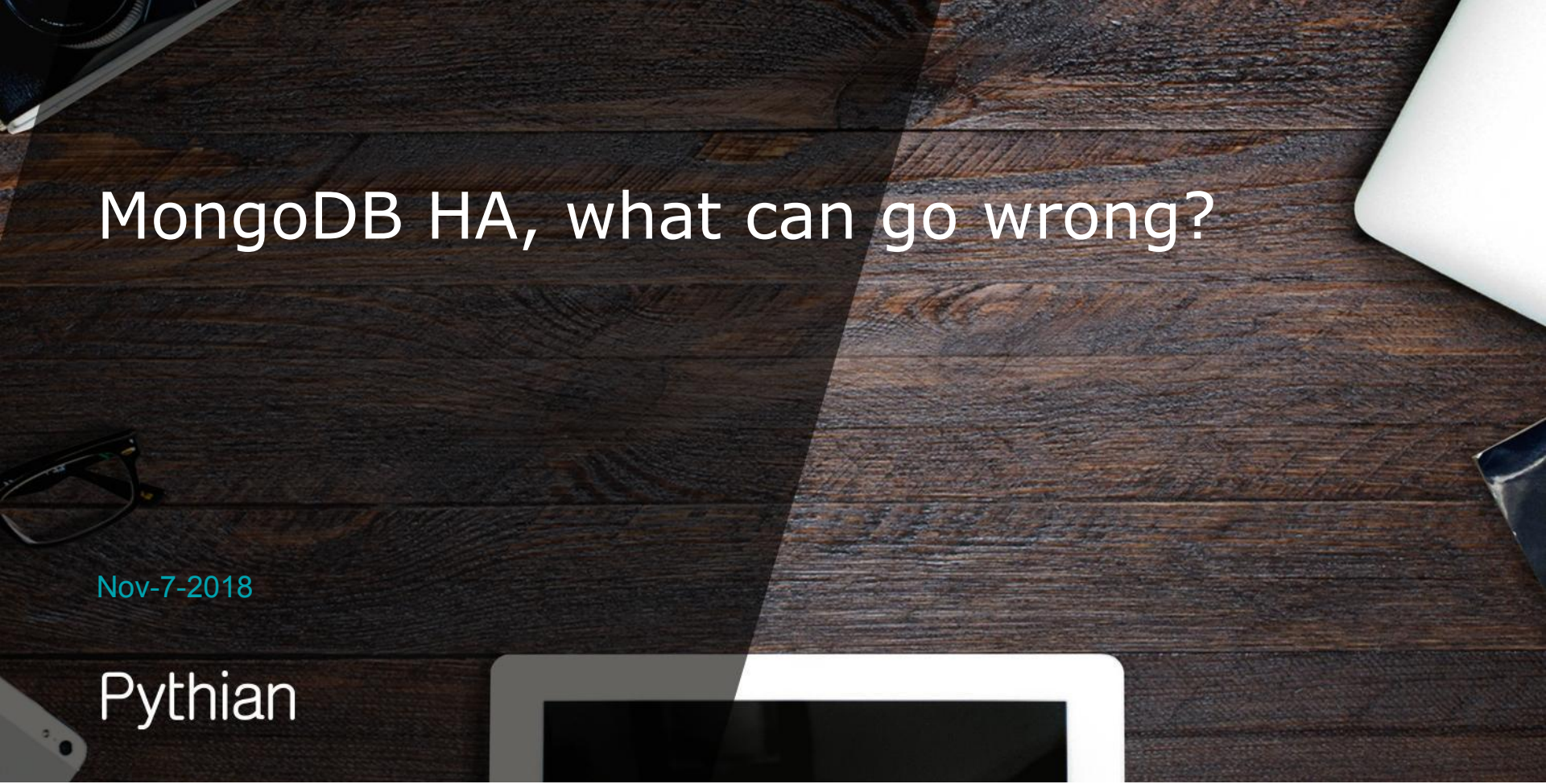


MongoDB HA, what can go wrong?

Nov-7-2018

Pythian

About me

- Location: Skopje, Republic of Macedonia
- Education: MSc, Software Engineering
- Experience:
  - Lead Database Consultant (since 2016)
  - Database Consultant (2012 - 2016)
  - Web Developer, DBA (2007 - 2012)
- Certifications: C100DBA - MongoDB certified DBA (since 2016)
- Percona speaker since 2016

https://mk.linkedin.com/in/igorle @igorle

Overview

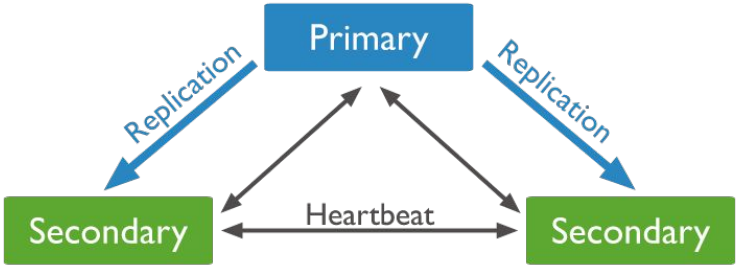
- What is replica set, how replication works
- Replication concept
- Replica set features, deployment architectures
- Hidden nodes, Arbiter nodes, Priority 0 nodes
- Production failures
- Monitoring replica set
- QA

Replication

Replica set

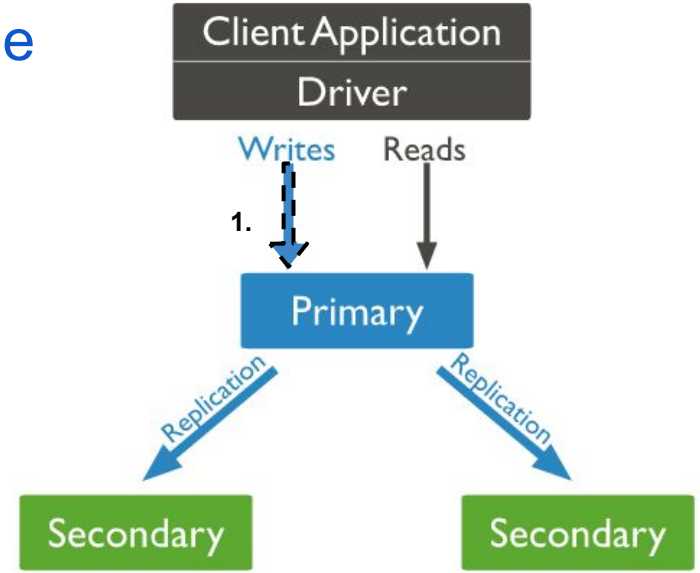
- Group of mongod processes that maintain the same data set
- Redundancy and high availability
- Increased read capacity (scaling reads)
- Automatic failover

# Members	# Nodes Required to Elect New Primary	Fault Tolerance
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3



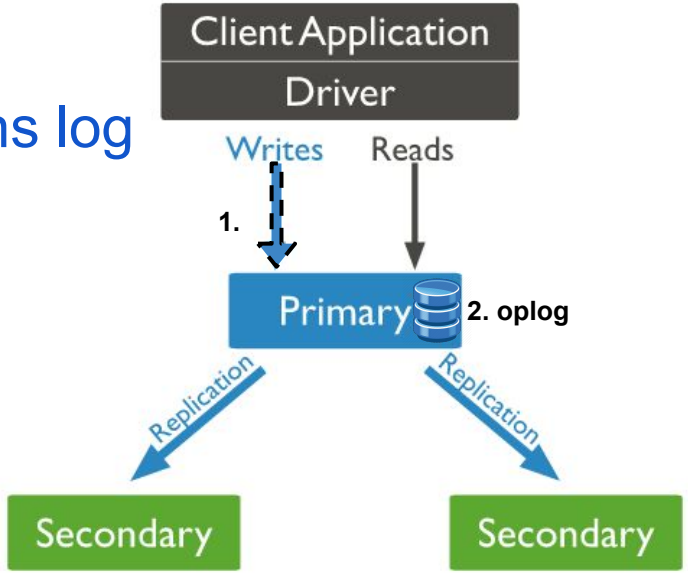
Replication concept

1. Write operations go to the Primary node
2. All changes are recorded into operations log
3. Asynchronous replication to Secondary
4. Secondaries copy the Primary oplog
5. Secondary can use sync source Secondary



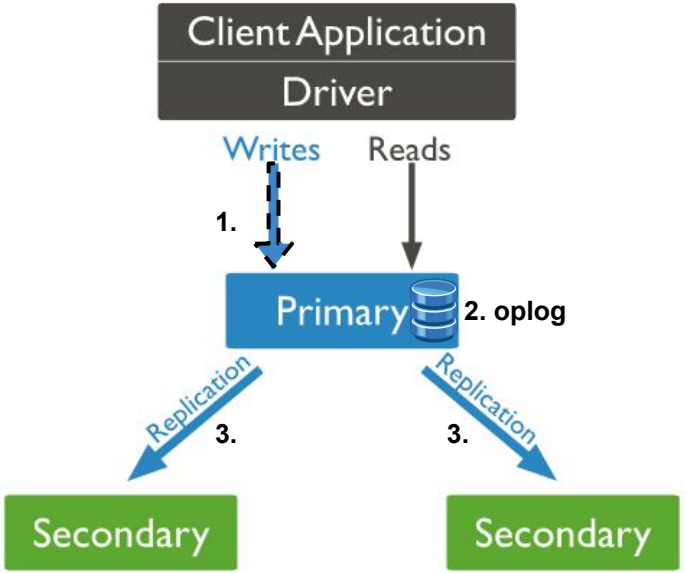
Replication concept

1. Write operations go to the Primary node
2. All changes are recorded into operations log
3. Asynchronous replication to Secondary
4. Secondaries copy the Primary oplog
5. Secondary can use sync source Secondary



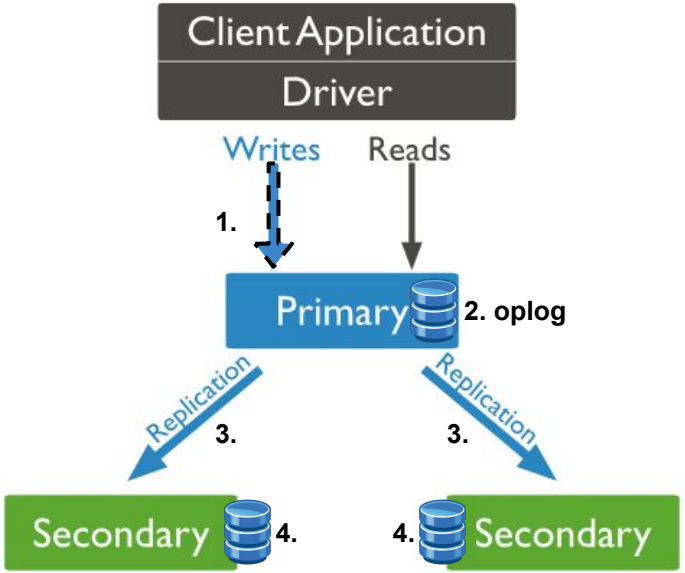
Replication concept

1. Write operations go to the Primary node
2. All changes are recorded into operations log
3. Asynchronous replication to Secondary
4. Secondaries copy the Primary oplog
5. Secondary can use sync source Secondary



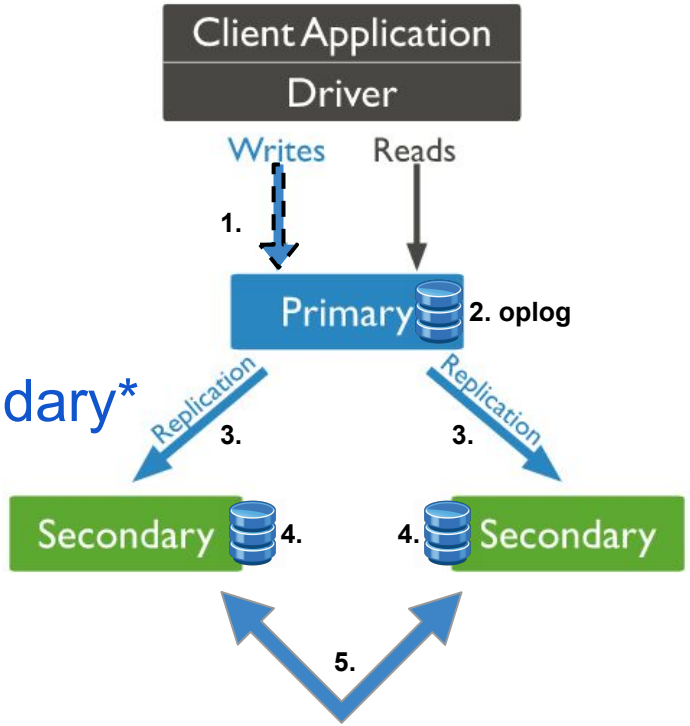
Replication concept

1. Write operations go to the Primary node
2. All changes are recorded into operations log
3. Asynchronous replication to Secondary
4. Secondaries copy the Primary oplog
5. Secondary can use sync source Secondary



Replication concept

1. Write operations go to the Primary node
2. All changes are recorded into operations log
3. Asynchronous replication to Secondary
4. Secondaries copy the Primary oplog
5. Secondary can use sync source Secondary*



*settings.chainingAllowed (true by default)

Replica set oplog

- Special capped collection that keeps a rolling record of all operations that modify the data stored in the databases
- Idempotent
- Default oplog size

For Unix and Windows systems

Storage Engine	Default Oplog Size	Lower Bound	Upper Bound
In-memory	5% of physical memory	50MB	50GB
WiredTiger	5% of free disk space	990MB	50GB
MMAPv1	5% of free disk space	990MB	50GB

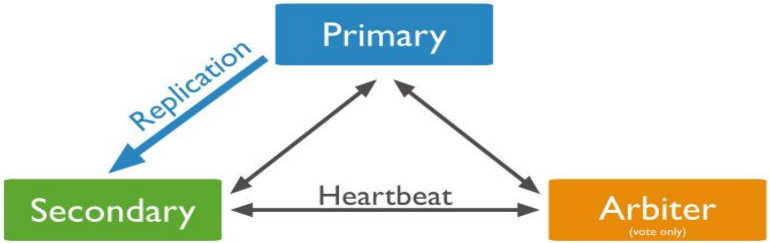
Configuration

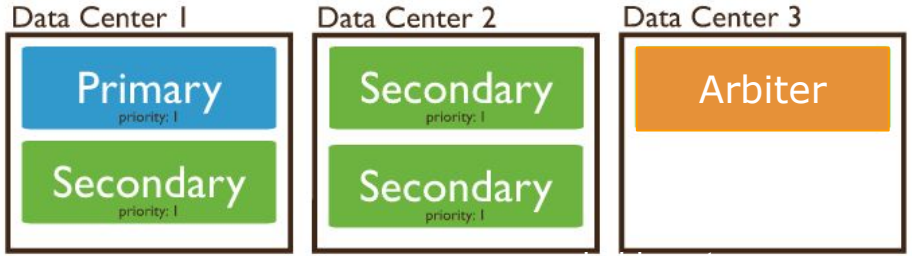
Configuration options

- 50 members per replica set (7 voting members)
- Arbiter node
- Priority 0 node
- Hidden node
- Delayed node

Arbiter node

- Does not hold copy of data
- Votes in elections

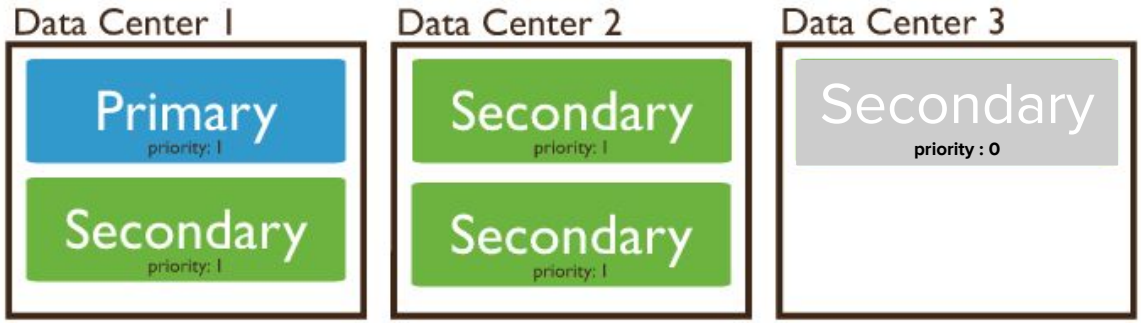




Priority 0 node

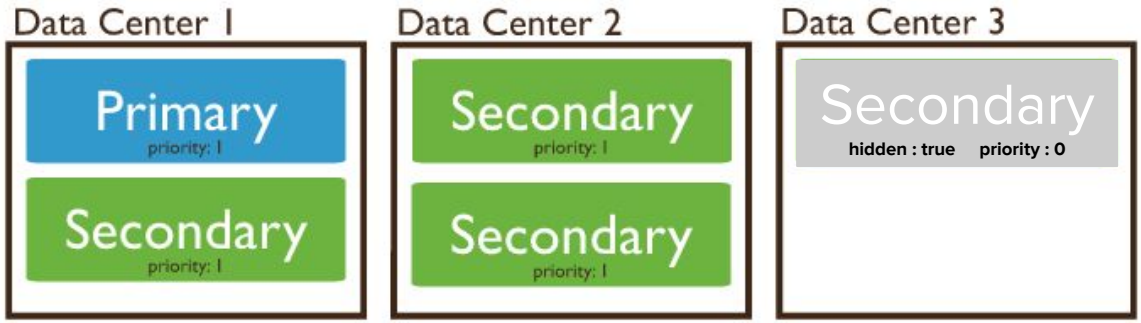
Priority - floating point (i.e. decimal) number between 0 and 1000

- Cannot become primary, cannot trigger election
- Visible to application (accepts reads/writes)
- Votes in elections



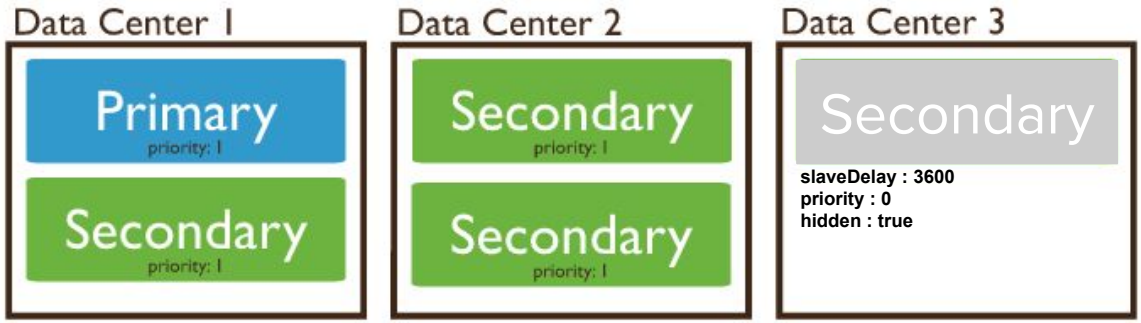
Hidden node

- Not visible to application
- Never becomes primary, but can vote in elections
- Use cases
  - reporting
  - backups



Delayed node

- Must be priority 0 member
- Should be hidden member (not mandatory)
- Mainly used for backups (historical snapshot of data)
- Recovery in case of human error

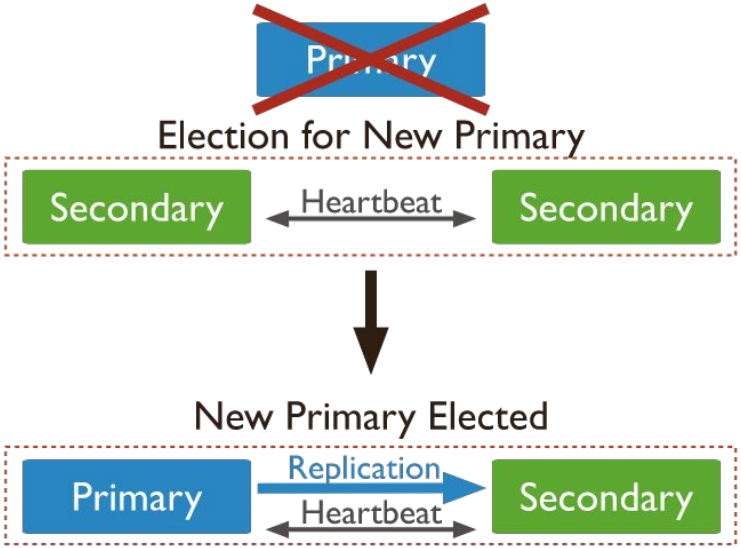


Failures

Small oplog size

1. Primary/Secondary node down
   - Node failure
   - Planned maintenance
2. Automatic Failover

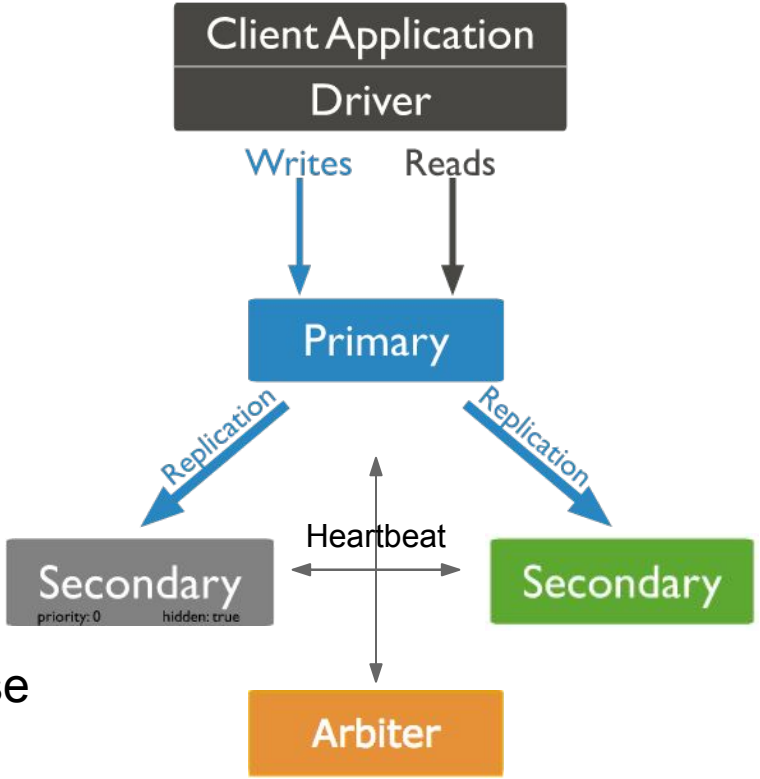
   …… (several hours later)
3. New Primary overwrites latest oplog
4. Failed Node needs resync



MongoDB >= 3.6: db.adminCommand({replSetResizeOplog: 1, size: 32000})

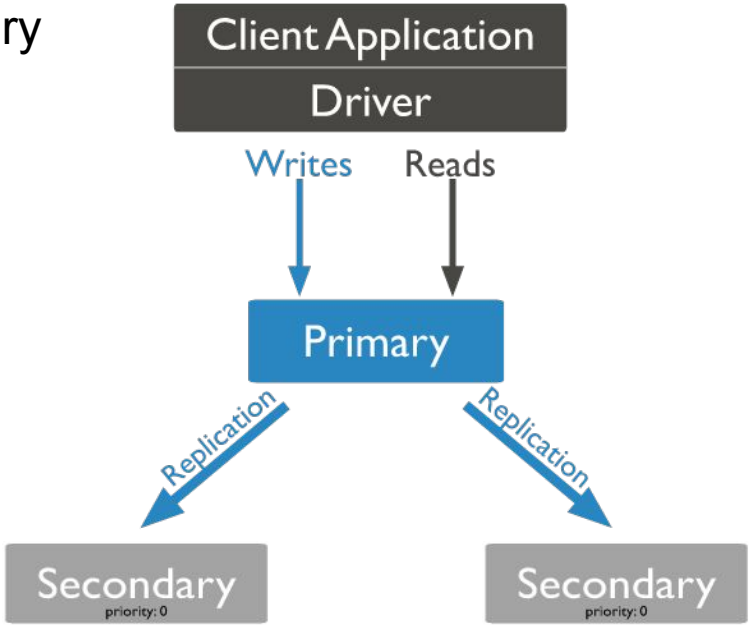
Arbiter nodes

- Votes in election
- Does not hold copy of data
- If 2 nodes are down, no majority to elect new Primary
- Fault tolerance is still 1 node
- 4 data nodes + 1 Arbiter makes more sense



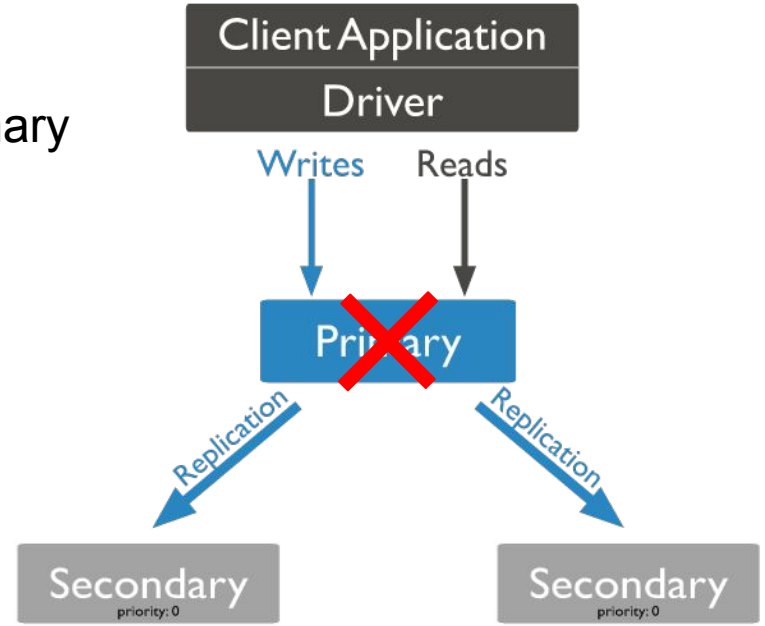
Priority 0 nodes

- Application driver sends writes to Primary
- Reads go to Primary by default
- Secondaries can serve reads
- Read preference
  - primary
  - primaryPreferred
  - secondary
  - secondaryPreferred
  - nearest



Priority 0 nodes

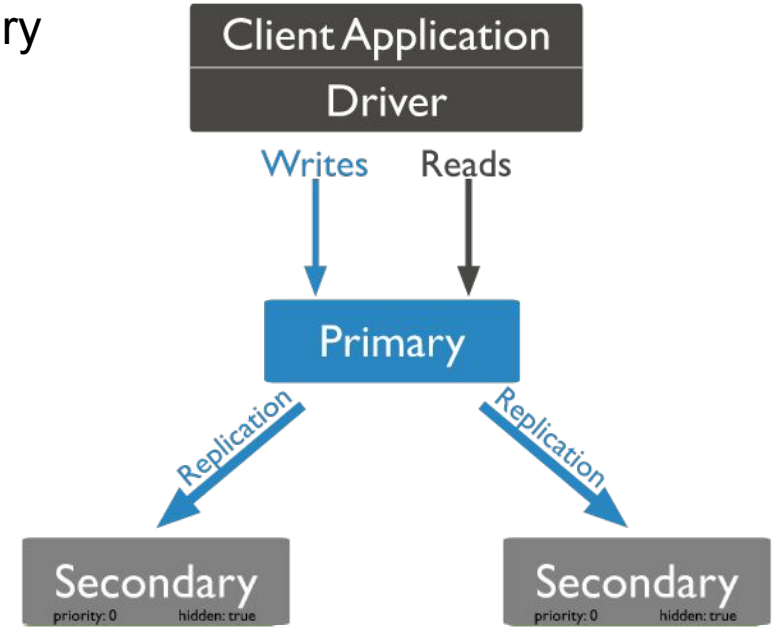
- Primary node fails
- Replica set starts election for new Primary
- Zero nodes eligible for Primary
- Application can not send writes
- Database is read only*



*depends on read preference setting

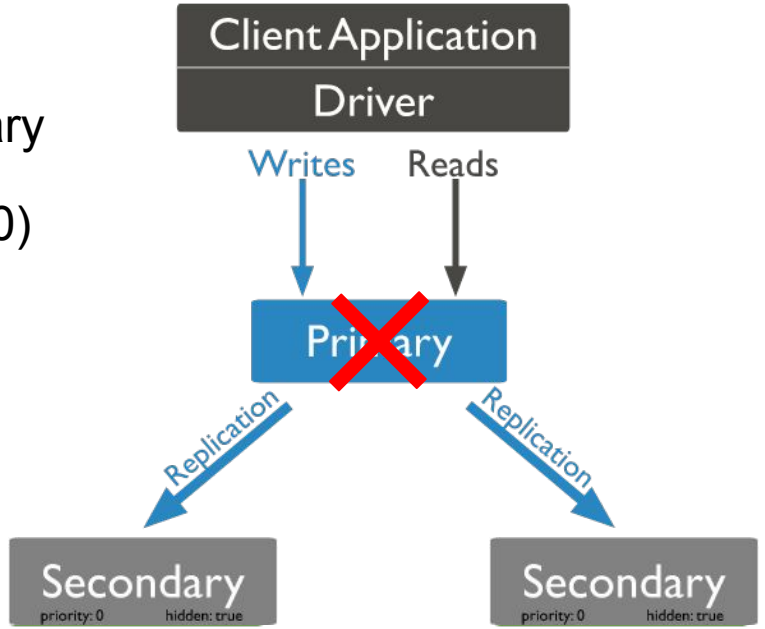
Hidden nodes

- Application driver sends writes to Primary
- Reads go to Primary by default
- Secondaries cannot serve reads
- Read preference
    - primary



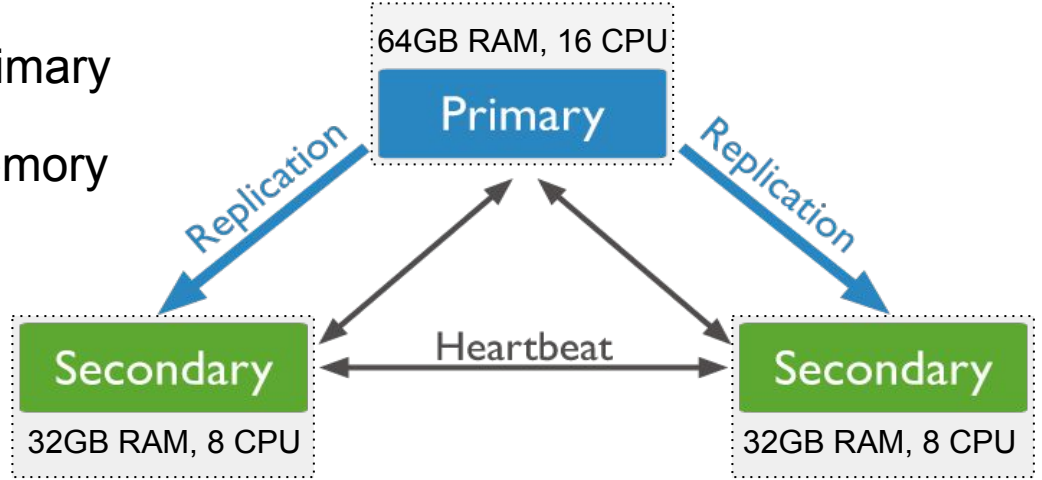
Hidden nodes

- Primary node fails
- Replica set starts election for new Primary
- Zero nodes eligible for Primary (priority:0)
- Application can not send writes/reads
- Downtime



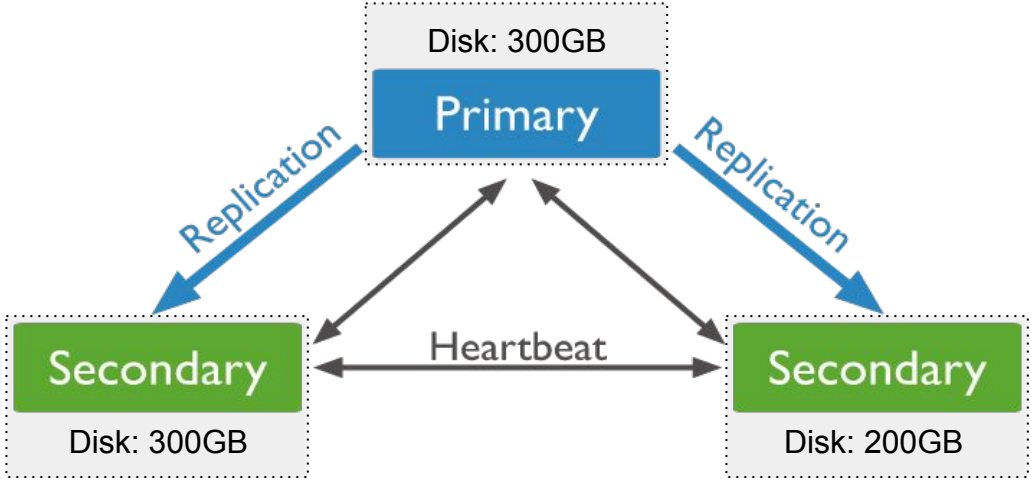
Hardware

- Primary node fails
- Secondary elected as new Primary
- Working set does not fit in memory
- Performance degradation
- Application stalls



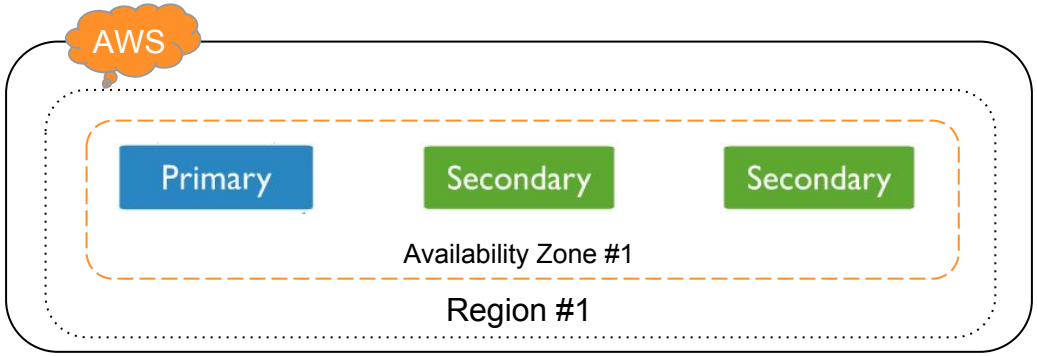
Hardware

- Dataset grows
- No Disk space on Secondary
- mongod process fail
- 2 nodes replica set
- Zero tolerance for failures



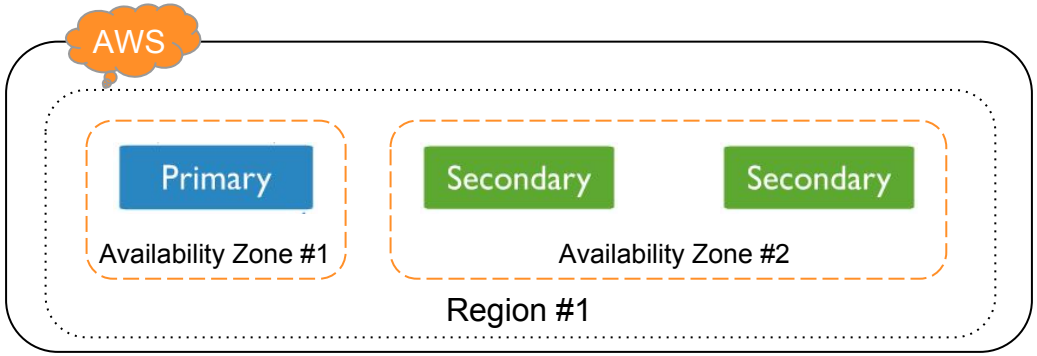
Cloud deployment

- All replica set members deployed in single Availability Zone
- Availability Zone #1 goes down
- Downtime



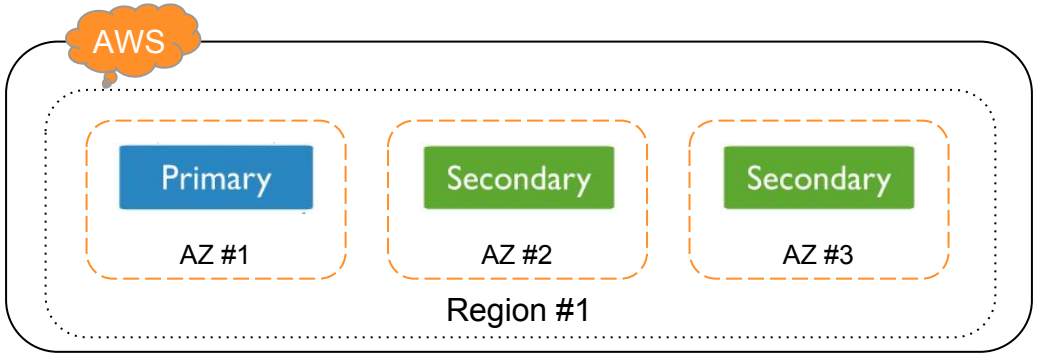
Cloud deployment

- Availability Zone #1 goes down
  - New Primary elected from AZ #2
- Availability Zone #2 goes down
  - Database is read only



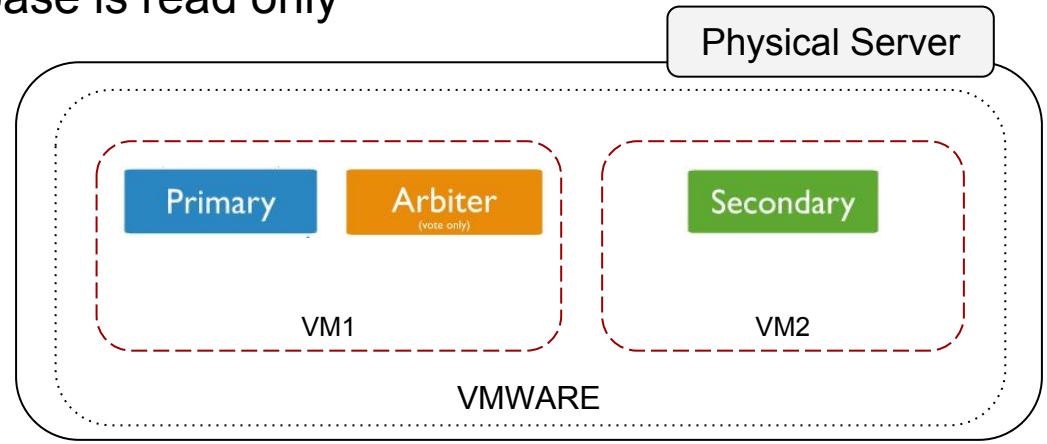
Cloud deployment

- Region #1 goes down
- Downtime



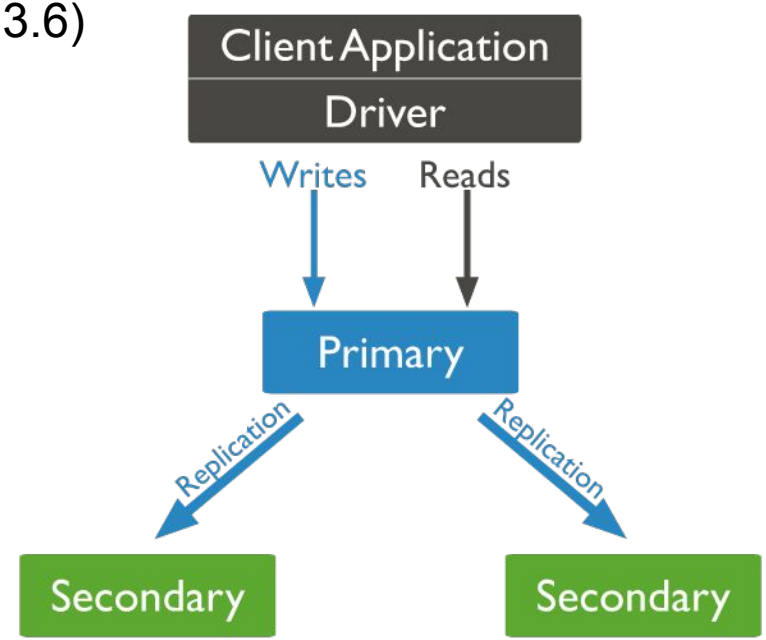
Virtualization

- VM2 goes down
  - Primary node has majority on VM1
- VM1 goes down
  - Database is read only



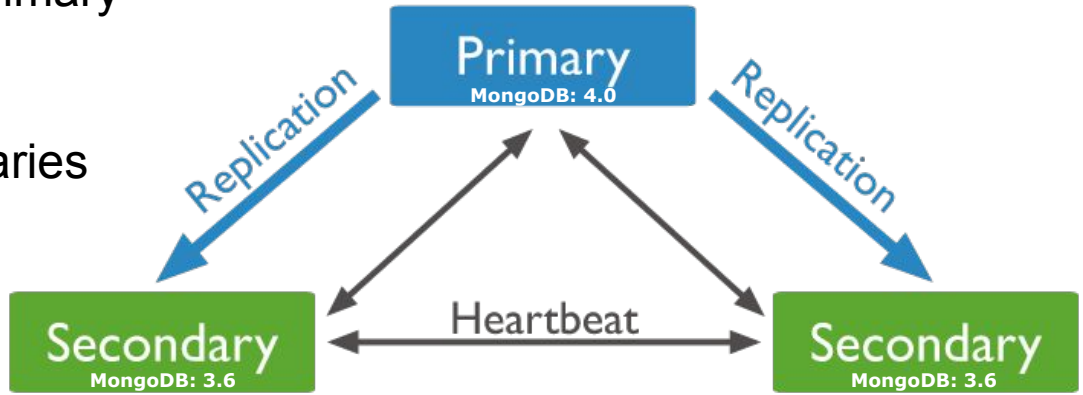
Version upgrades

- Replica set major version upgrade (3.4 > 3.6)
- Driver v3.4 not compatible with DB v3.6
- Application cannot send requests
- Downtime
- Rollback to previous DB version



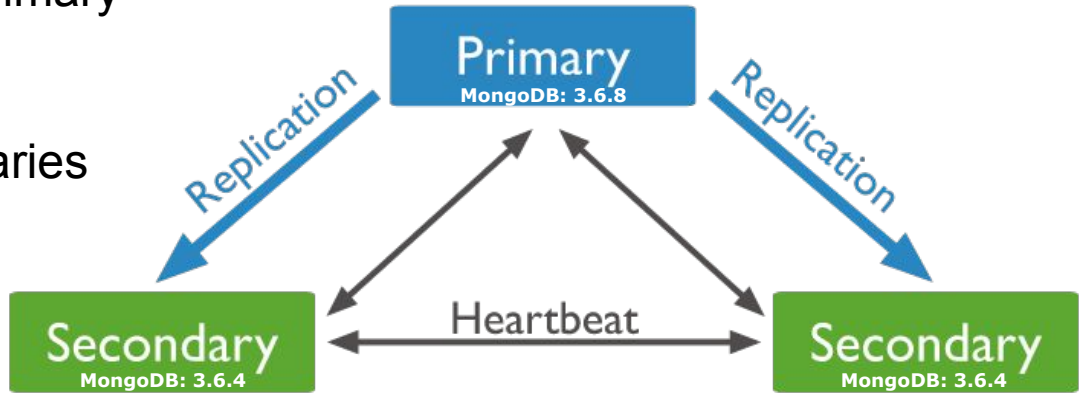
Version upgrades

- Replica set major version upgrade
- Promote new version as Primary
- Confirm application works
- Forget to upgrade Secondaries
- Start using new features
- New Primary elected
- Application errors

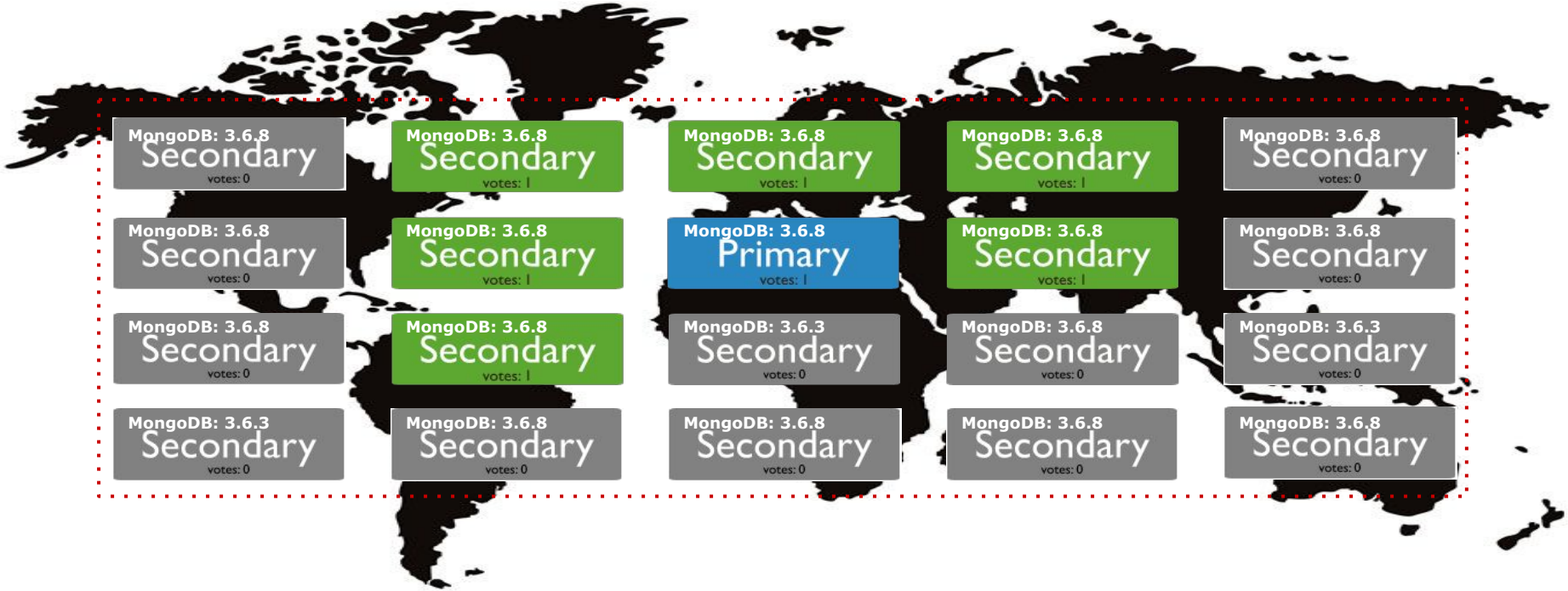


Version upgrades

- Minor version upgrade
- Promote new version as Primary
- Confirm application works
- Forget to upgrade Secondaries
- Bug fixes in minor release
- New Primary elected
- Application errors

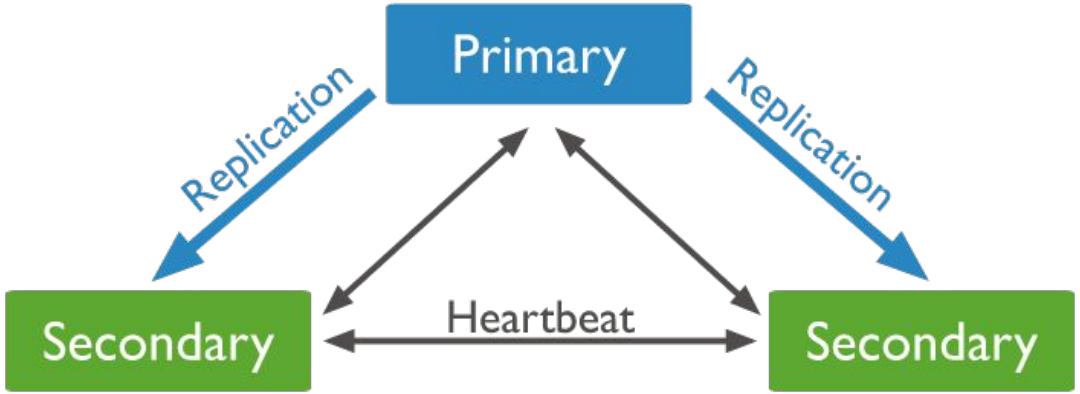


Version upgrades



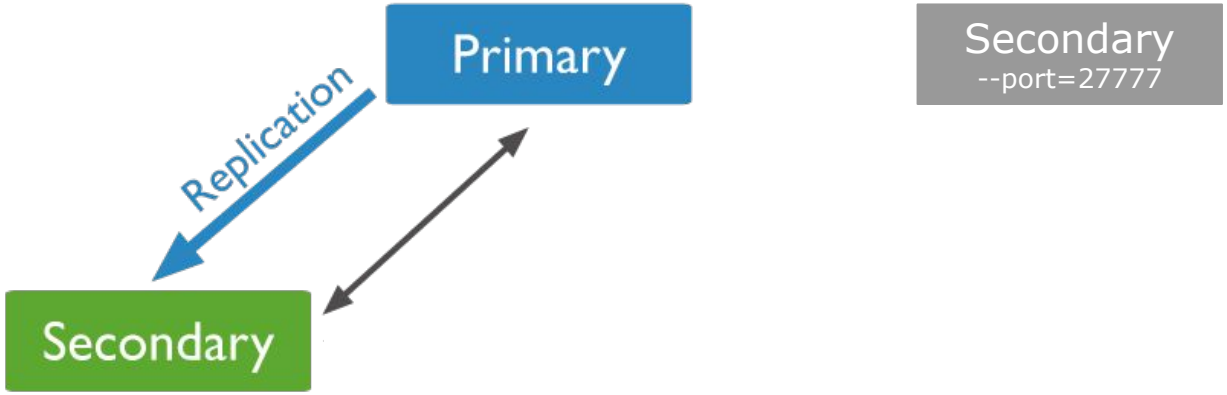
DDL operation

- Adding index on a collection
- Connect to the Primary node
    - db.people.createIndex({ zipcode: 1 }, { background: true })



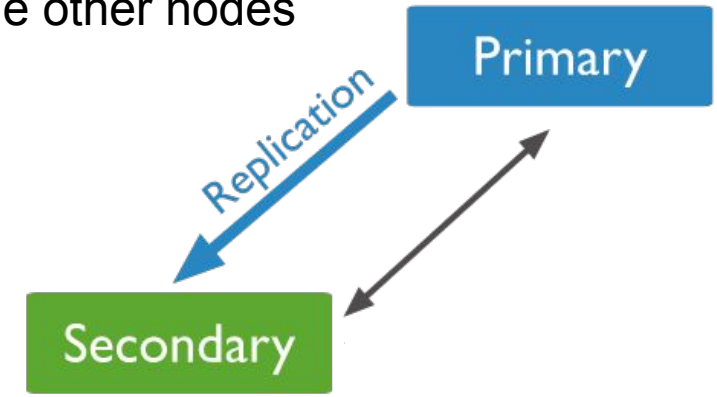
DDL operation

- Stop one Secondary
- Restart on different port



DDL operation

- Add the Index
- Rejoin to replica
- Promote Secondary as Primary
- Forget the other nodes

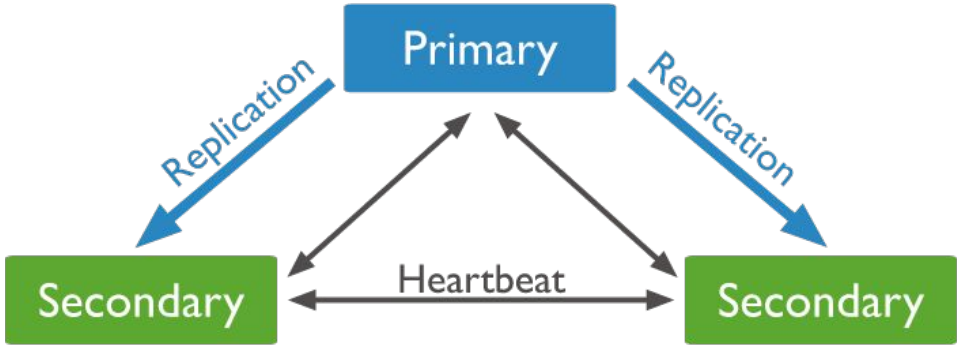


Secondary
--port=27777

db.people.createIndex({zipcode:1})

Backups

- Pick one Secondary
- db.fsyncLock()
- Take snapshot
- db.fsyncUnlock()
- Unlock fails
- Secondary starts lagging
- Primary overwrites oplog
- Secondary needs initial sync



Monitoring replica set

- Replica set has no Primary
- Number of unhealthy members is above threshold
- Replication lag is above threshold
- Replica set elected new Primary
- Host of any type has restarted
- Host of type Secondary is recovering
- Host of any type is down
- Host of any time has experienced Rollback
- Monitoring backup status

Summary

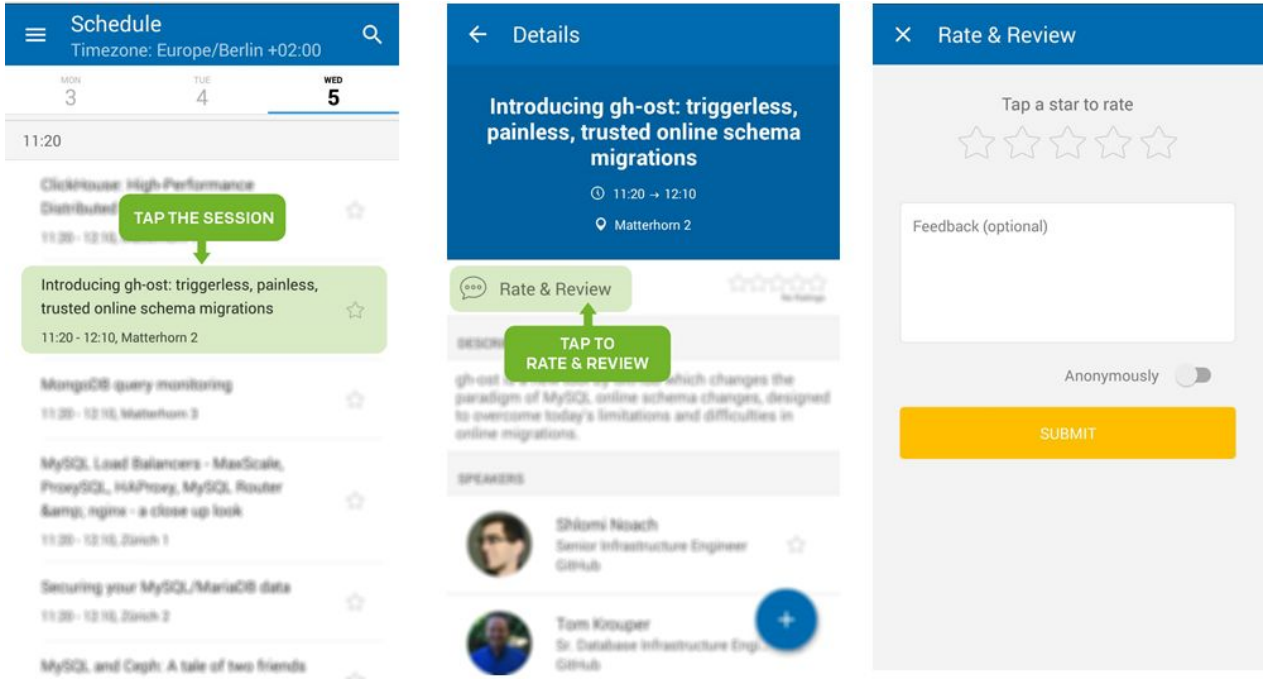
- Replica set with odd number of voting members
- Hidden or Delayed member for dedicated functions (reporting, backups …)
- Have more than one eligible Primary in the replica set
- Use multi-AZ for Cloud deployments
- Don’t deploy more than one mongod process per node/host
- Run replica set members with same hardware for all nodes
- Run replica set members with same mongo version
- Monitor your replica set status and nodes
- Monitor replication lag and Oplog size

Questions?

We’re Hiring!

https://www.pythian.com/careers/

Rate my session



Pythian
love your data