MongoDB HA, what can go wrong?
About Me

{"name": "Igor Donchovski",
"live_in": "Skopje",
"email": "donchovski@pythian.com",
"current_role": "Lead database consultant",
"education": [{"type": "College", "name": "FEIT", "graduated": "2008", "university": "UKIM"},
               {"type": "Master", "name": "FINKI", "graduated": "2013", "university": "UKIM"}],
"work": [{"role": "Web developer", "start": "2007", "end": "2012", "company": "Gord Systems"},
         {"role": "DBA", "start": "2012", "end": "2014", "company": "NOVP"},
         {"role": "Database consultant", "start": "2014", "end": "2016", "company": "Pythian"},
         {"role": "Lead database consultant", "start": "2016", "company": "Pythian"}],
"certificates": [{"name": "C100DBA", "year": "2016", "description": "MongoDB certified DBA"}],
"social": [{"network": "LinkedIn", "url": "https://mk.linkedin.com/in/igorle"},
           {"network": "Twitter", "url": "https://twitter.com/igorle"}],
"interests": ["Hiking", "Biking", "Traveling"],
"hobbies": ["Painting", "Photography", "Cooking"],
"proud_of": ["Volunteering", "Helping the Community"]}
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

<table>
<thead>
<tr>
<th># Members</th>
<th># Nodes Required to Elect New Primary</th>
<th>Fault Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
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

© 2019 Pythian. Confidential
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

<table>
<thead>
<tr>
<th>Storage Engine</th>
<th>Default Oplog Size</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-memory</td>
<td>5% of physical memory</td>
<td>50MB</td>
<td>50GB</td>
</tr>
<tr>
<td>WiredTiger</td>
<td>5% of free disk space</td>
<td>990MB</td>
<td>50GB</td>
</tr>
<tr>
<td>MMAPv1</td>
<td>5% of free disk space</td>
<td>990MB</td>
<td>50GB</td>
</tr>
</tbody>
</table>
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
Everyone on the same page?
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 (default)
  - 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 fails
- 2 nodes replica set
- Zero tolerance for failures
Network

- Heartbeat lost
- Primary step down
- New Primary election
- Application timeout*
- Rollback

Best Practice: Test Primary step down for your application

*Retryable writes since MongoDB 3.6
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 (3.6>4.0)
- Driver v3.6 not compatible with DB v4.0
- Compatibility changes
- 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
Sharded Clusters

App Server
Router (mongos)

1 or more Routers

Config Servers (replica set)

2 or more Shards
Shard (replica set)
Sharded Clusters

App Server

Router (mongos)

1 or more Routers

Config Servers (replica set)

2 or more Shards

Shard (replica set)

Shard (replica set)
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 type has experienced Rollback
- Network issues between members of the replica set or cluster
- 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/