

Replication



- ❑ Replication provides redundancy and increases data availability.
- ❑ With multiple copies of data on different database servers, replication protects a database from the loss of a single server.
- ❑ Replication also allows you to recover from hardware failure and service interruptions.
- ❑ With additional copies of the data, you can dedicate one to disaster recovery, reporting, or backup.
- ❑ In some cases, you can use replication to increase read capacity.
- ❑ Clients have the ability to send read and write operations to different servers.
- ❑ You can also maintain copies in different data centers to increase the locality and availability of data for distributed applications.

Replicaiton in MongoDB

- A replica set is a group of mongod instances that host the same data set.
- One mongod, the primary, receives all write operations.
- All other instances, secondaries, apply operations from the primary so that they have the same data set.
- The primary accepts all write operations from clients.
- Replica set can have only one primary.
- Because only one member can accept write operations, replica sets provide strict consistency.
- To support replication, the primary logs all changes to its data sets in its oplog.
- The secondaries replicate the primary's oplog and apply the operations to their data sets.
- Secondaries' data sets reflect the primary's data set.
- If the primary is unavailable, the replica set will elect a secondary to be primary.
- By default, clients read from the primary, however, clients can specify a read preferences to send read operations to secondaries.

Arbiter

- ❑ You may add an extra mongod instance a replica set as an arbiter.
- ❑ Arbiters do not maintain a data set.
- ❑ Arbiters only exist to vote in elections.
- ❑ If your replica set has an even number of members, add an arbiter to obtain a majority of votes in an election for primary.
- ❑ Arbiters do not require dedicated hardware.

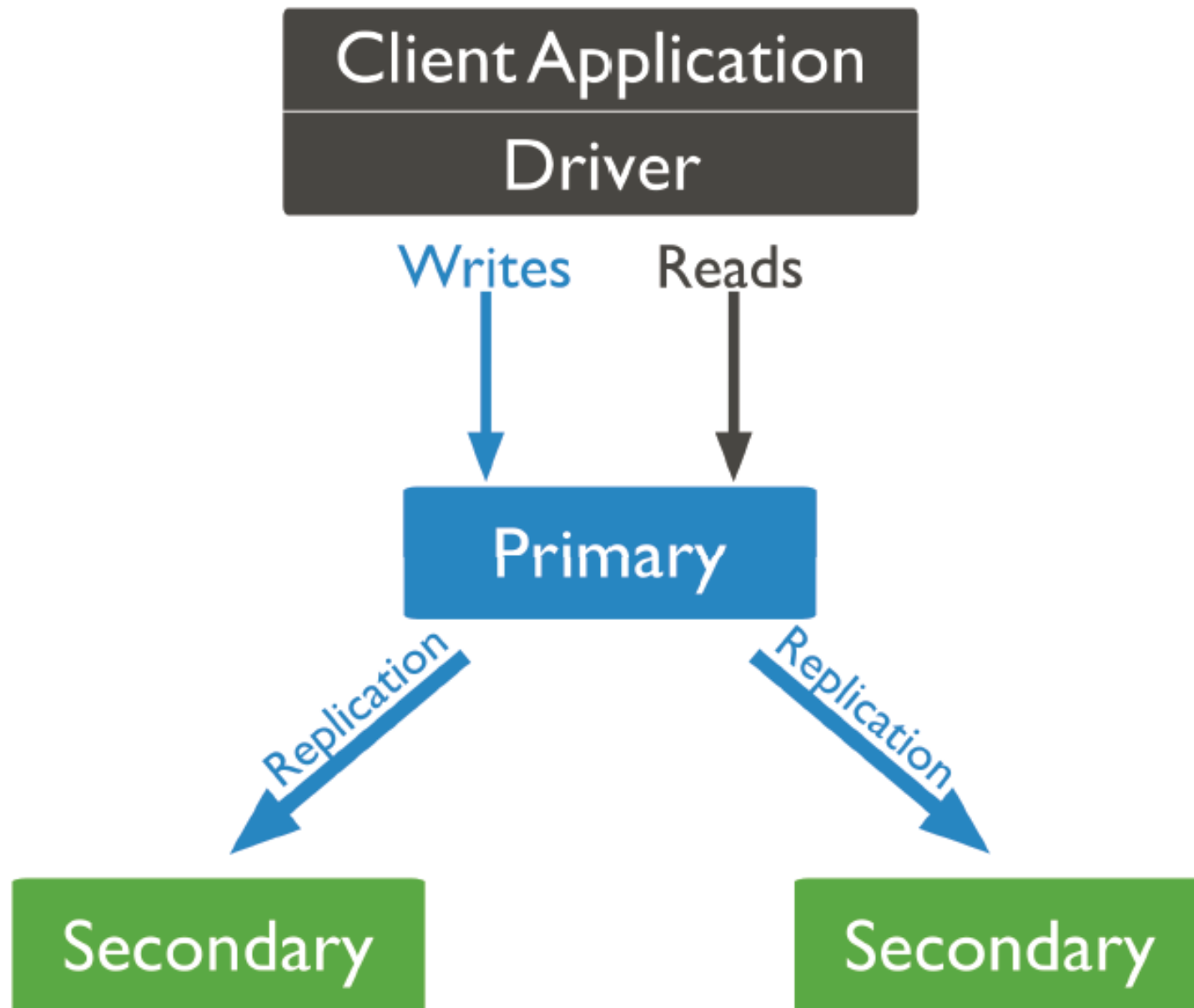


Diagram of default routing of reads and writes to the primary.

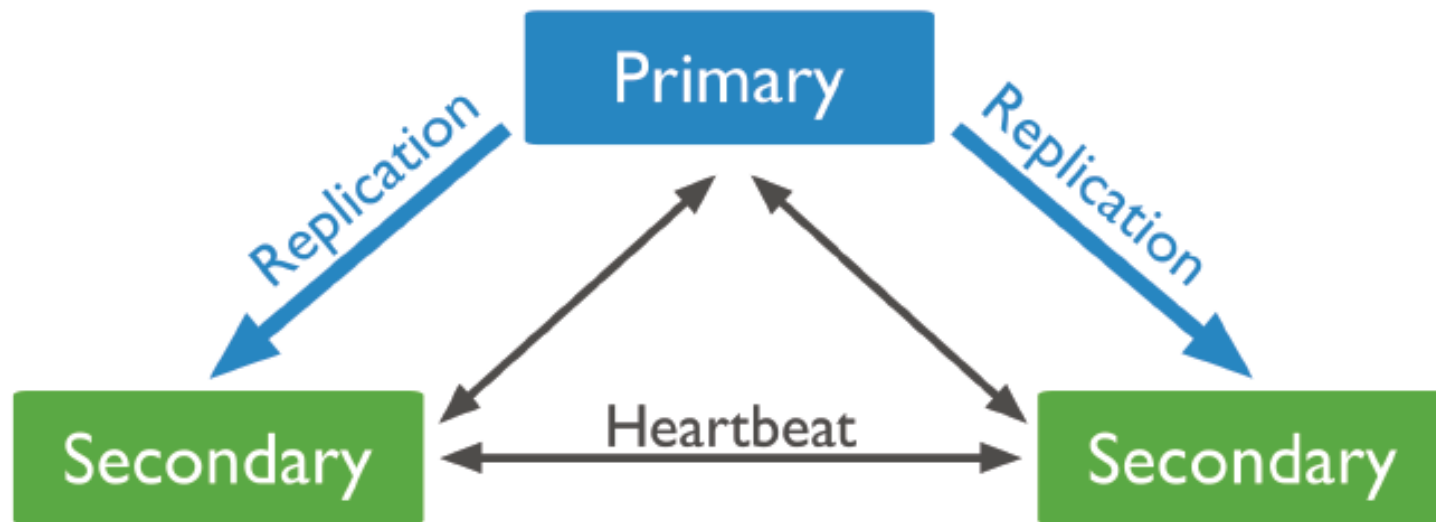


Diagram of a 3 member replica set that consists of a primary and two secondaries.

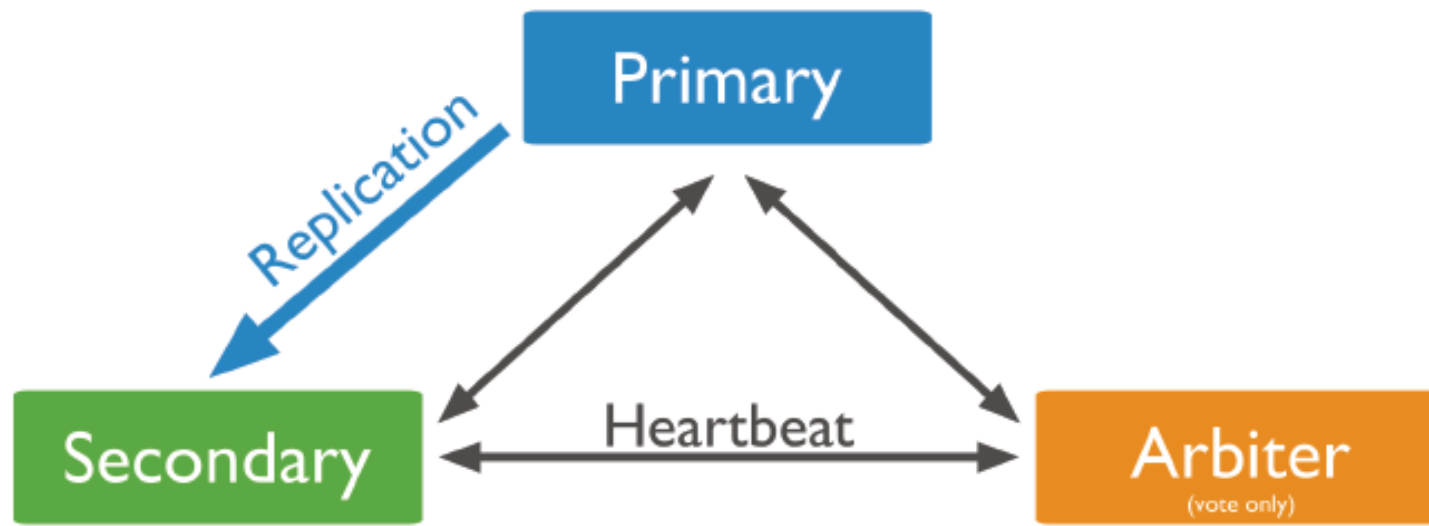



Diagram of 3 member replica set that consists of a primary, a secondary, and an arbiter.

Asynchronous Replication

- 
- Secondaries apply operations from the primary asynchronously.
 - By applying operations after the primary, sets can continue to function without some members.
 - However, as a result secondaries may not return the most current data to clients.

Automatic Failover



- When a primary does not communicate with the other members of the set for more than 10 seconds, the replica set will attempt to select another member to become the new primary.
- The first secondary that receives a majority of the votes becomes primary.

Additional Features



- Replica sets provide a number of options to support application needs.
- For example, you may deploy a replica set with members in multiple data centers, or control the outcome of elections by adjusting the priority of some members.
- Replica sets also support dedicated members for reporting, disaster recovery, or backup functions.

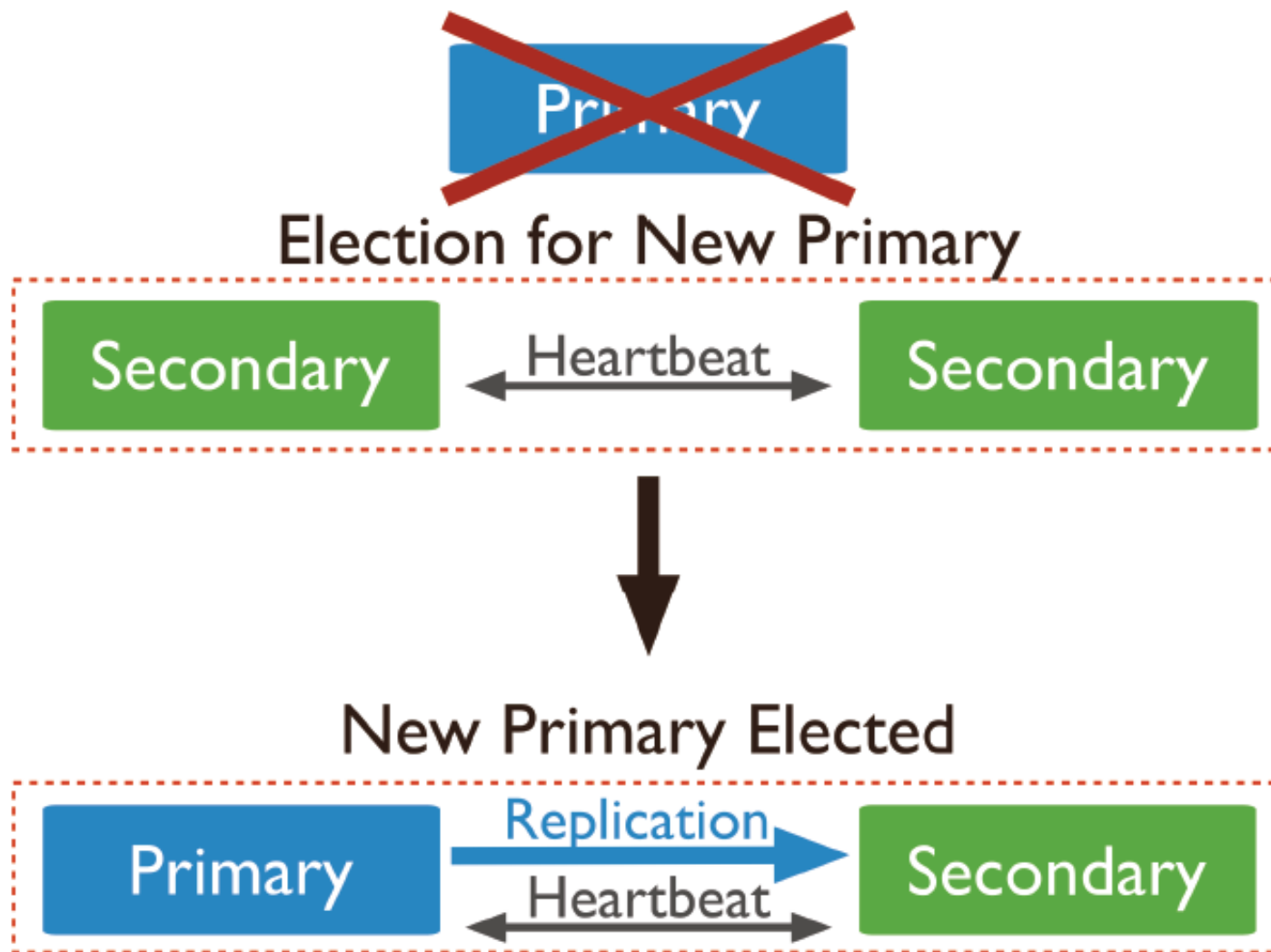


Diagram of an election of a new primary. In a three member replica set with two secondaries, the primary becomes unreachable. The loss of a primary triggers an election where one of the secondaries becomes the new primary

Replica Set Members

- ❑ A replica set in MongoDB is a group of mongod processes that provide redundancy and high availability.
- ❑ The members of a replica set are:
 - ❑ Primary (receives all write operations)
 - ❑ Secondaries (replicate operations from the primary to maintain an identical data set)
 - Secondaries may have additional configurations for special usage profiles.
 - For example, secondaries may be nonvoting or priority 0.
- ❑ You can also maintain an arbiter as part of a replica set.
- ❑ Arbiters do not keep a copy of the data.
- ❑ However, arbiters play a role in the elections that select a primary if the current primary is unavailable.
- ❑ A replica set can have up to 12 members. However, only 7 members can vote at a time.
- ❑ The minimum requirements for a replica set are:
 - ❑ A primary - a secondary - an arbiter
- ❑ Most deployments, however, will keep three members that store data: A primary and two secondary members

Replica Set Primary

- ❑ The primary is the only member in the replica set that receives write operations.
- ❑ MongoDB applies write operations on the primary and then records the operations on the primary's oplog.
- ❑ Secondary members replicate this log and apply the operations to their data sets.
- ❑ In the following three-member replica set, the primary accepts all write operations.
- ❑ Then the secondaries replicate the oplog to apply to their data sets.
- ❑ All members of the replica set can accept read operations.
- ❑ However, by default, an application directs its read operations to the primary member.
- ❑ The replica set can have at most one primary.
- ❑ If the current primary becomes unavailable, an election determines the new primary.
- ❑ In the following 3-member replica set, the primary becomes unavailable.
- ❑ This triggers an election which selects one of the remaining secondaries as the new primary.

Replica Set Secondary Members

- ❑ A secondary maintains a copy of the primary's data set.
- ❑ To replicate data, a secondary applies operations from the primary's oplog to its own data set in an asynchronous process.
- ❑ A replica set can have one or more secondaries.
- ❑ The following three-member replica set has two secondary members.
- ❑ Secondaries replicate the primary's oplog & apply the operations to their data sets.
- ❑ Although clients cannot write data to secondaries, clients can read data from secondary members.
- ❑ A secondary can become a primary.
- ❑ If the current primary becomes unavailable, the replica set holds an election to choose with of the secondaries becomes the new primary.
- ❑ In the following three-member replica set, the primary becomes unavailable.
- ❑ This triggers an election where one of the remaining secondaries becomes the new primary.

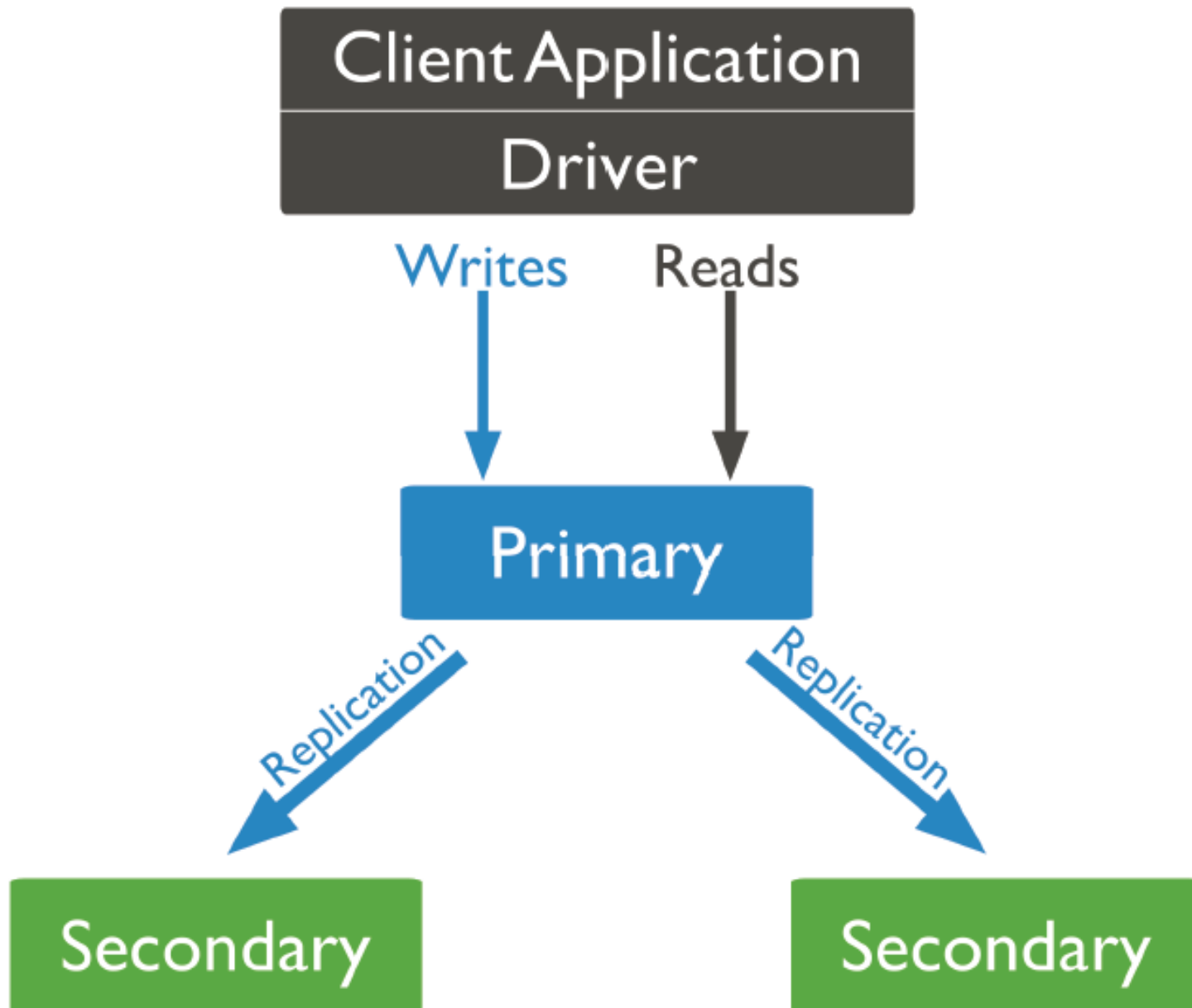


Diagram of default routing of reads and writes to the primary.

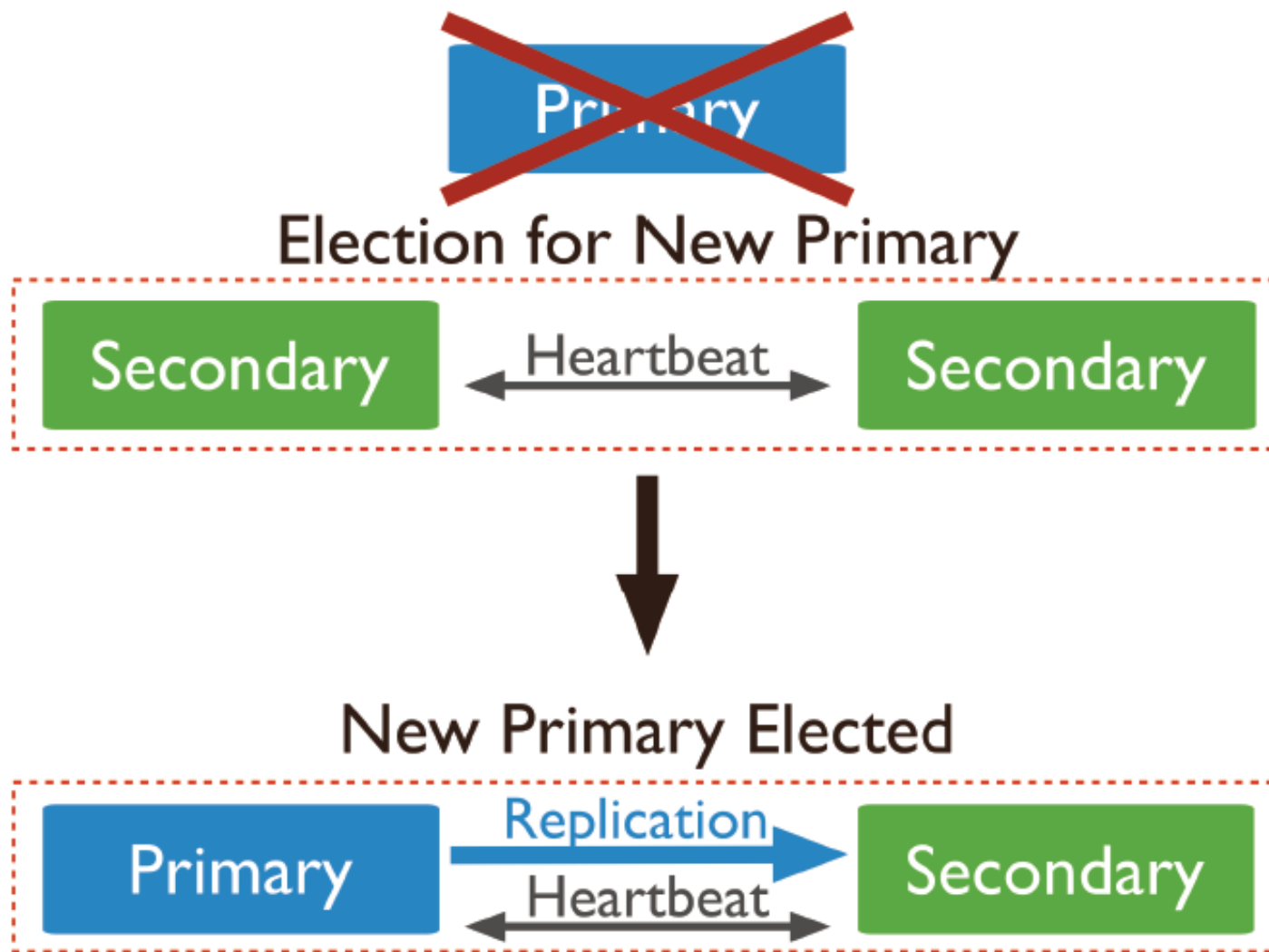


Diagram of an election of a new primary. In a three member replica set with two secondaries, the primary becomes unreachable. The loss of a primary triggers an election where one of the secondaries becomes the new primary

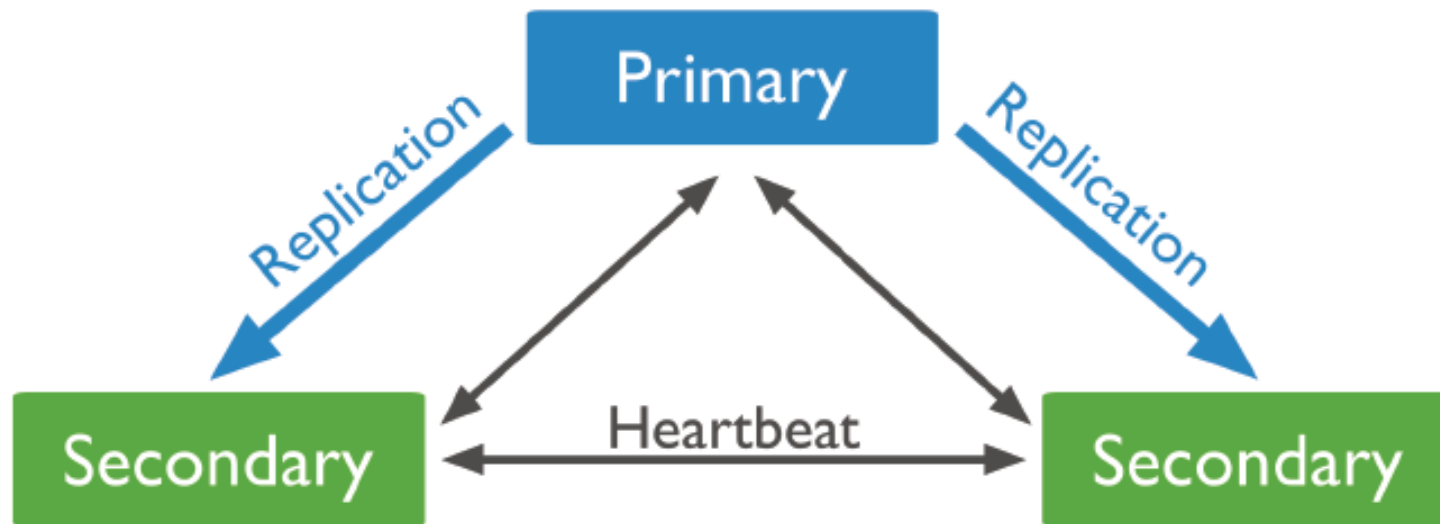


Diagram of a 3 member replica set that consists of a primary and two secondaries.

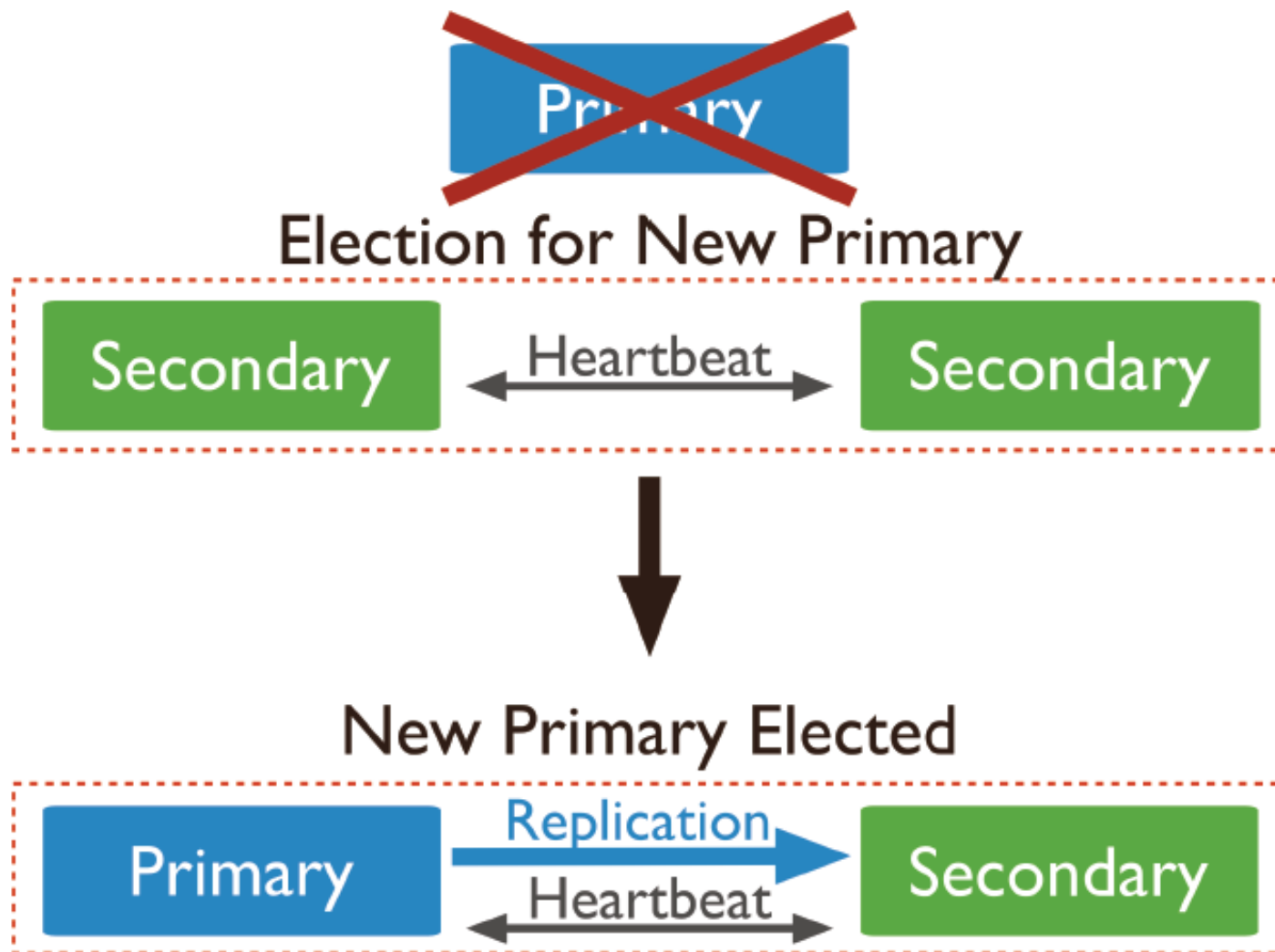


Diagram of an election of a new primary. In a three member replica set with two secondaries, the primary becomes unreachable. The loss of a primary triggers an election where one of the secondaries becomes the new primary

A secondary member can be configured for a specific purpose

- You can configure a secondary to:
 - ▣ Priority 0 Replica Set Members
 - Prevent it from becoming a primary in an election, which allows it to reside in a secondary data center or to serve as a cold standby.
 - ▣ Hidden Replica Set Members
 - Prevent applications from reading from it, which allows it to run applications that require separation from normal traffic.
 - ▣ Delayed Replica Set Members
 - Keep a running “historical” snapshot for use in recovery from certain errors, such as unintentionally deleted databases.

Priority 0 Replica Set Members

- A priority 0 member is a secondary that cannot become primary, and cannot trigger elections.
- Otherwise these members function as normal secondaries.
- A priority 0 member maintains a copy of the data set, accepts read operations, and votes in elections.
- Configure a priority 0 member to prevent secondaries from becoming primary, which is particularly useful in multi-data center deployments.
- In a three-member replica set, in one data center hosts the primary and a secondary.
- A second data center hosts one priority 0 member that cannot become primary.
- A priority 0 member can function as a standby. In some replica sets, it might not be possible to add a new member in a reasonable amount of time.
- A standby member keeps a current copy of the data to be able to replace an unavailable member.

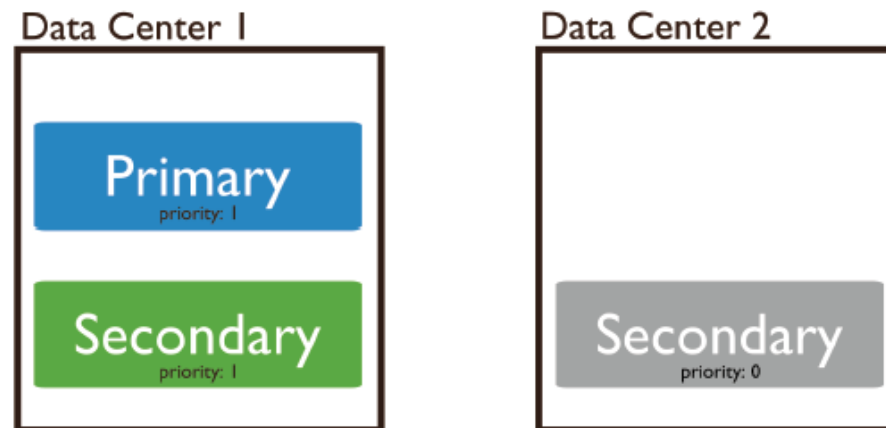


Diagram of a 3 member replica set distributed across two data centers. Replica set includes a priority 0 member.

Hidden Replica Set Members

- ❑ A hidden member maintains a copy of the primary's data set but is invisible to client applications.
- ❑ Hidden members are ideal for workloads with different usage patterns from the other members in the replica set.
- ❑ Hidden members are also priority 0 members and cannot become primary.
- ❑ The `db.isMaster()` method does not display hidden members.
- ❑ Hidden members, however, do vote in elections.
- ❑ In the following five-member replica set, all four secondary members have copies of the primary's data set, but one of secondary members is hidden.
- ❑ Secondary reads do not reach a hidden member, so the member receives no traffic beyond what replication requires.
- ❑ Use hidden members for instances that require separation from normal traffic.
- ❑ For example, you might use hidden members for instances dedicated to reporting or to backups.
- ❑ For dedicated backup, ensure that the hidden member has low network latency to network to the primary or likely primary.
- ❑ Ensure that the replication lag is minimal or non-existent.

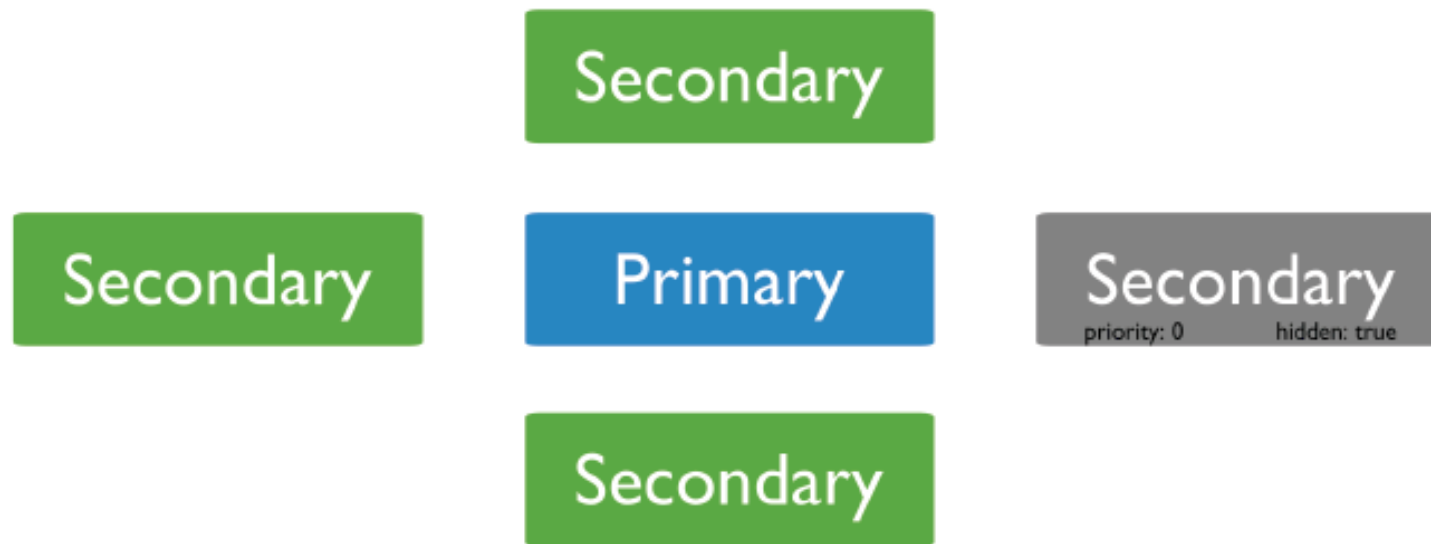


Diagram of a 5 member replica set with a hidden priority 0 member.

Delayed Replica Set Members

- Delayed members contain copies of a replica set's data set.
- However, a delayed member's data set reflects an earlier, or delayed, state of the set.
- For example, if the current time is 09:52 and a member has a delay of an hour, the delayed member has no operation more recent than 08:52.
- Because delayed members are a “rolling backup” or a running “historical” snapshot of the data set, they may help
- you recover from various kinds of human error.
- For example, a delayed member can make it possible to recover from unsuccessful application upgrades and operator errors including dropped databases and collections.

Requirements for Delayed members:

- Must be priority 0 members.
 - ▣ Set the priority to 0 to prevent a delayed member from becoming primary.
- Should be hidden members.
 - ▣ Always prevent applications from seeing and querying delayed members.
- do vote in elections for primary.
 - ▣ Delayed members apply operations from the oplog on a delay.
- When choosing the amount of delay, consider that the amount of delay:
 - ▣ must be is equal to or greater than your maintenance windows.
 - ▣ must be smaller than the capacity of the oplog.

Example

In the following 5-member replica set, the primary and all secondaries have copies of the data set. One member applies operations with a delay of 3600 seconds, or an hour. This delayed member is also *hidden* and is a *priority 0 member*.

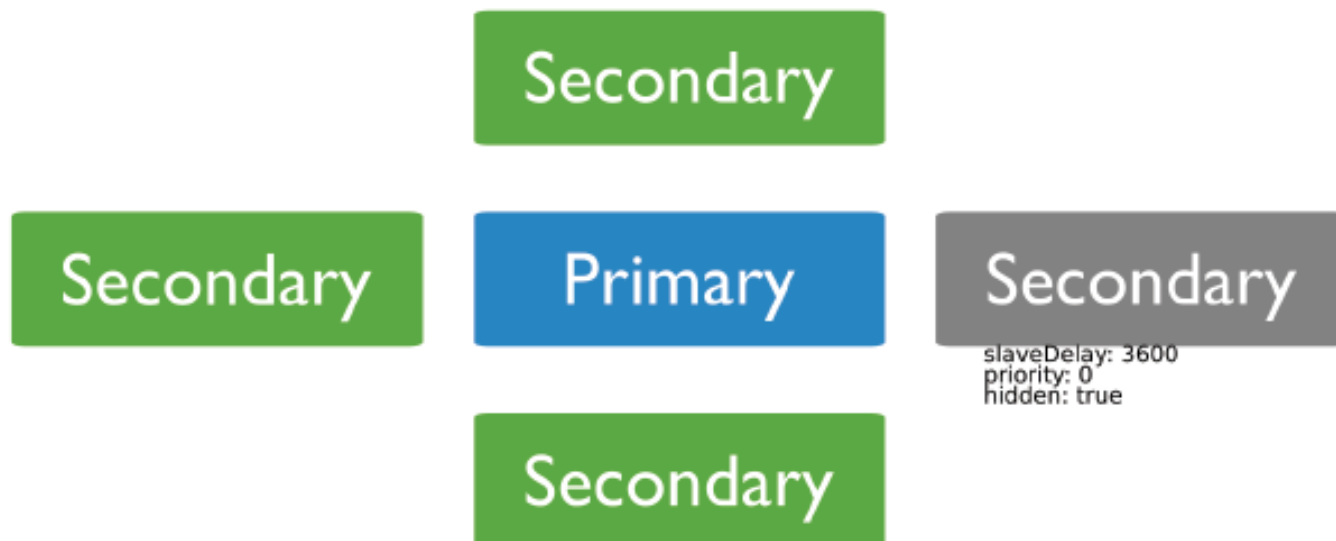


Diagram of a 5 member replica set with a hidden delayed priority 0 member.

Arbiter

- An arbiter does not have a copy of data set and cannot become a primary.
- Replica sets may have arbiters to add a vote in elections of for primary.
- Arbiters allow replica sets to have an uneven number of members, without the overhead of a member that replicates data.
- Important: Do not run an arbiter on systems that also host the primary or the secondary members of the replica set.
- Only add an arbiter to sets with even numbers of members.
- If you add an arbiter to a set with an odd number of members, the set may suffer from tied elections.

Example

For example, in the following replica set, an arbiter allows the set to have an odd number of votes for elections:

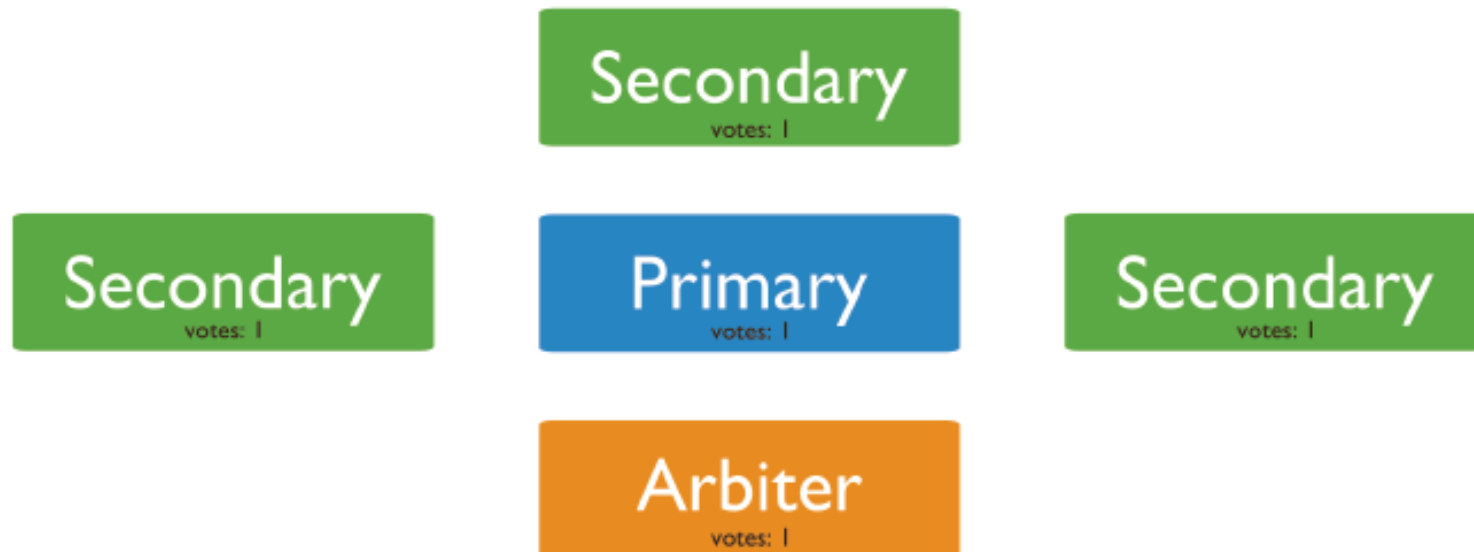


Diagram of a four member replica set plus an arbiter for odd number of votes.

Replica Set Deployment Architectures



- The standard replica set deployment for production system is a three-member replica set.
- These sets provide redundancy and fault tolerance.
- Avoid complexity when possible, but let your application requirements dictate the architecture.

Strategies

Determine the Number of Members

Add members in a replica set according to these strategies.

Deploy an Odd Number of Members

An odd number of members ensures that the replica set is always able to elect a primary. If you have an even number of members, add an arbiter to get an odd number. *Arbiters* do not store a copy of the data and require fewer resources. As a result, you may run an arbiter on an application server or other shared process.

Consider Fault Tolerance

Fault tolerance for a replica set is the number of members that can become unavailable and still leave enough members in the set to elect a primary. In other words, it is the difference between the number of members in the set and the majority needed to elect a primary. Without a primary, a replica set cannot accept write operations. Fault tolerance is an effect of replica set size, but the relationship is not direct. See the following table:

Number of Members.	Majority Required to Elect a New Primary.	Fault Tolerance.
3	2	1
4	3	1
5	3	2
6	4	2

Adding a member to the replica set does not *always* increase the fault tolerance. However, in these cases, additional members can provide support for dedicated functions, such as backups or reporting.

Use Hidden and Delayed Members for Dedicated Functions

to support dedicated functions, such as backup or reporting.

Load Balance on Read-Heavy Deployments

In a deployment with *very* high read traffic, you can improve read throughput by distributing reads to secondary members. As your deployment grows, add or move members to alternate data centers to improve redundancy and availability.

Always ensure that the main facility is able to elect a primary.

Add Capacity Ahead of Demand

The existing members of a replica set must have spare capacity to support adding a new member. Always add new members before the current demand saturates the capacity of the set.

Determine the Distribution of Members

Distribute Members Geographically

To protect your data if your main data center fails, keep at least one member in an alternate data center. Set these members' `priority` to 0 to prevent them from becoming primary.

Keep a Majority of Members in One Location

When a replica set has members in multiple data centers, network partitions can prevent communication between data centers. To replicate data, members must be able to communicate to other members.

In an election, members must see each other to create a majority. To ensure that the replica set members can confirm a majority and elect a primary, keep a majority of the set's members in one location.

Target Operations with Tags

Use `replica set tags` to ensure that operations replicate to specific data centers. Tags also support targeting read operations to specific machines.

Use Journaling to Protect Against Power Failures

Enable journaling to protect data against service interruptions. Without journaling MongoDB cannot recover data after unexpected shutdowns, including power failures and unexpected reboots.

All 64-bit versions of MongoDB after version 2.0 have journaling enabled by default.