***Replica Set High Availability* (page 580)** Presents the details of the automatic failover and recovery process with replica sets.

***Replica Set Elections* (page 580)** Elections occur when the primary becomes unavailable and the replica set members autonomously select a new primary.

***Read Preference* (page 588)** Applications specify *read preference* to control how drivers direct read operations to members of the replica set.

***Replication Processes* (page 593)** Mechanics of the replication process and related topics.

***Master Slave Replication* (page 596)** Master-slave replication provided redundancy in early versions of MongoDB. Replica sets replace master-slave for most use cases.

9.2.1 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* (page ??).** The primary receives all write operations.

***Secondaries* (page ??).** 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 *non-voting* (page 583) or *priority 0* (page 567).

You can also maintain an *arbiter* (page **??**) 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.

The minimum requirements for a replica set are: A *primary* (page **??**), a *secondary* (page **??**), and an *arbiter* (page **??**). Most deployments, however, will keep three members that store data: A *primary* (page **??**) and two *secondary members* (page **??**).

Changed in version 3.0.0: A replica set can have up to *50 members* (page 800) but only 7 voting members. [4] In previous versions, replica sets can have up to 12 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* (page 593). *Secondary* (page **??**) 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. See *Read Preference* (page 588) for details on changing the default read behavior.

The replica set can have at most one primary. [5] If the current primary becomes unavailable, an election determines the new primary. See *Replica Set Elections* (page 580) for more details.

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.

[4] While replica sets are the recommended solution for production, a replica set can support up to `50 members` in total. If your deployment requires more than 50 members, you'll need to use *master-slave* (page 596) replication. However, master-slave replication lacks the automatic failover capabilities.

[5] In some circumstances, two nodes in a replica set may *transiently* believe that they are the primary, but at most, one of them will be able to complete writes with *{w: majority} write concern* (page 130). The node that can complete *{w: majority}* (page 130) writes is the current primary, and the other node is a former primary that has not yet recognized its demotion, typically due to a *network partition*. When this occurs, clients that connect to the former primary may observe stale data despite having requested read preference `primary` (page 657).