- The `repairDatabase` command converts indexes to a 2.0 indexes.

To convert all indexes for a given collection to the *2.0 type* (page 910), invoke the `compact` command.

Once you create new indexes, downgrading to 1.8.x will require a re-index of any indexes created using 2.0. See *Build Old Style Indexes* (page 523).

Sharding Authentication

Applications can now use authentication with *sharded clusters*.

Replica Sets

Hidden Nodes in Sharded Clusters In 2.0, `mongos` instances can now determine when a member of a replica set becomes "hidden" without requiring a restart. In 1.8, `mongos` if you reconfigured a member as hidden, you *had* to restart `mongos` to prevent queries from reaching the hidden member.

Priorities Each *replica set* member can now have a priority value consisting of a floating-point from 0 to 1000, inclusive. Priorities let you control which member of the set you prefer to have as *primary* the member with the highest priority that can see a majority of the set will be elected primary.

For example, suppose you have a replica set with three members, `A`, `B`, and `C`, and suppose that their priorities are set as follows:

- `A`'s priority is `2`.
- `B`'s priority is `3`.
- `C`'s priority is `1`.

During normal operation, the set will always chose `B` as primary. If `B` becomes unavailable, the set will elect `A` as primary.

For more information, see the `priority` documentation.

Data-Center Awareness You can now "tag" *replica set* members to indicate their location. You can use these tags to design custom *write rules* (page 78) across data centers, racks, specific servers, or any other architecture choice.

For example, an administrator can define rules such as "very important write" or `customerData` or "audit-trail" to replicate to certain servers, racks, data centers, etc. Then in the application code, the developer would say:

```
db.foo.insert(doc, {w : "very important write"})
```

which would succeed if it fulfilled the conditions the DBA defined for "very important write".

For more information, see *Data Center Awareness* (page 207).

Drivers may also support tag-aware reads. Instead of specifying `slaveOk`, you specify `slaveOk` with tags indicating which data-centers to read from. For details, see the Drivers[885] documentation.

`w : majority` You can also set `w` to `majority` to ensure that the write propagates to a majority of nodes, effectively committing it. The value for "majority" will automatically adjust as you add or remove nodes from the set.

For more information, see *Write Concern* (page 78).

[885] https://docs.mongodb.org/ecosystem/drivers