Config Server Availability

If one or two config servers become unavailable, the cluster's metadata becomes *read only*. You can still read and write data from the shards, but no chunk migrations or splits will occur until all three servers are available.

If all three config servers are unavailable, you can still use the cluster if you do not restart the `mongos` instances until after the config servers are accessible again. If you restart the `mongos` instances before the config servers are available, the `mongos` will be unable to route reads and writes.

Clusters become inoperable without the cluster metadata. To ensure that the config servers remain available and intact, backups of config servers are critical. The data on the config server is small compared to the data stored in a cluster, and the config server has a relatively low activity load. These properties facilitate finding a window to back up the config servers.

If the name or address that a sharded cluster uses to connect to a config server changes, you must restart **every** `mongod` and `mongos` instance in the sharded cluster. Avoid downtime by using CNAMEs to identify config servers within the MongoDB deployment.

See *Renaming Config Servers and Cluster Availability* (page 677) for more information.

10.2.2 Sharded Cluster Architectures

The following documents introduce deployment patterns for sharded clusters.

Sharded Cluster Requirements

While sharding is a powerful and compelling feature, sharded clusters have significant infrastructure requirements and increases the overall complexity of a deployment. As a result, only deploy sharded clusters when indicated by application and operational requirements

Sharding is the *only* solution for some classes of deployments. Use *sharded clusters* if:

- your data set approaches or exceeds the storage capacity of a single MongoDB instance.
- the size of your system's active *working set will soon* exceed the capacity of your system's *maximum* RAM.
- a single MongoDB instance cannot meet the demands of your write operations, and all other approaches have not reduced contention.

If these attributes are not present in your system, sharding will only add complexity to your system without adding much benefit.

Important: It takes time and resources to deploy sharding. If your system has *already* reached or exceeded its capacity, it will be difficult to deploy sharding without impacting your application.

As a result, if you think you will need to partition your database in the future, **do not** wait until your system is over capacity to enable sharding.

When designing your data model, take into consideration your sharding needs.