MongoDB supported *tag-aware sharding* from even the initial versions of MongoDB. This means tagging a range of shard keys values, associating that range with a shard, and redirecting operations to that specific tagged shard. This tag-aware sharding, since version 3.4, is referred to as ZONES. So, the only change is its name, and this is the reason sh.addShardTag(shard, tag) method is being used.

How it works

1. With the help of a shard key, MongoDB allows you to create zones of sharded data – also known as shard zones.
2. Each zone can be associated with one or more shards.
3. Similarly, a shard can associate with any number of non-conflicting zones.
4. MongoDB migrates chunks to the zone range in the selected shards.
5. MongoDB routes read and write to a particular zone range that resides in particular shards.

Useful for what kind of deployments/applications?

1. In cases where data needs to be routed to a particular shard due to some hardware configuration restrictions.
2. Zones can be useful if there is the need to isolate specific data to a particular shard. For example, in the case of GDPR compliance that requires businesses to protect data and privacy for an individual within the EU.
3. If an application is being used geographically and you want a query to route to the nearest shards for both reads and writes.

Let’s consider a Scenario

Consider the scenario of a school where students are experts in Biology, but most students are experts in Maths. So we have more data for the maths students compare to Biology students. In this example, deployment requires that Maths students data should route to the shard with the better configuration for a large amount of data. Both read and write will be served by specific shards.  All the Biology students will be served by another shard. To implement this, we will add a tag to deploy the zones to the shards.

For this scenario we have an environment with:

DB:**“school”**

Collection: **“students”**

Fields: **“sId”, “subject”, “marks”** and so on..

Indexed Fields: **“subject”** and **“sId”**

We enable sharding:

|  |  |
| --- | --- |
| 1 | sh.enableSharding("school") |

And create a shardkey: **“subject”** and **“sId”**

|  |  |
| --- | --- |
| 1 | sh.shardCollection("school.students", {subject: 1, sId: 1}); |

We have two shards in our test environment

**shards:**

|  |  |
| --- | --- |
| 1  2 | {  "\_id" : "shard0000",  "host" : "127.0.0.1:27001",  "state" : 1 }  {  "\_id" : "shard0001",  "host" : "127.0.0.1:27002",  "state" : 1 } |

Zone Deployment

**1) Disable balancer**

To prevent migration of the chunks across the cluster, disable the balancer for the “**students”** collection:

|  |  |
| --- | --- |
| 1  2 | mongos> sh.disableBalancing("school.students")  WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) |

Before proceeding further make sure the balancer is not running. It is not a mandatory process, but it is always a good practice to make sure no migration of chunks takes place while configuring zones

|  |  |
| --- | --- |
| 1  2 | mongos> sh.isBalancerRunning()  false |

**2) Add shard to the zone**

A zone can be associated with a particular shard in the form of a tag, using the sh.addShardTag(), so a tag will be added to each shard. Here we are considering two zones so the tags “MATHS” and “BIOLOGY” need to be added.

|  |  |
| --- | --- |
| 1  2  3  4 | mongos> sh.addShardTag( "shard0000" , "MATHS");  { "ok" : 1 }  mongos> sh.addShardTag( "shard0001" , "BIOLOGY");  { "ok" : 1 } |

We can see zones are assigned in the form of tags as required against each shard.

|  |  |
| --- | --- |
| 1  2  3  4 | mongos> sh.status()  shards:          {  "\_id" : "shard0000",  "host" : "127.0.0.1:27001",  "state" : 1,  "tags" : [ "MATHS" ] }          {  "\_id" : "shard0001",  "host" : "127.0.0.1:27002",  "state" : 1,  "tags" : [ "BIOLOGY" ] } |

**3) Define ranges for each zone**

Each zone covers one or more ranges of [shard key](https://docs.mongodb.com/v3.4/reference/glossary/#term-shard-key) values. Note: each range a zone covers is always inclusive of its lower boundary and exclusive of its upper boundary.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | mongos> sh.addTagRange(  "school.students",  { "subject" : "maths", "sId" : MinKey},  { "subject" : "maths", "sId" : MaxKey},  "MATHS"  )  { "ok" : 1 }  mongos> sh.addTagRange(  "school.students",  { "subject" : "biology", "sId" : MinKey},  { "subject" : "biology", "sId" : MaxKey},  "BIOLOGY"  )  { "ok" : 1 } |

**4) Enable balancer**

Now enable the balancer so the chunks will migrate across the shards as per the requirement and all the read and write queries will be routed to the particular shards.

|  |  |
| --- | --- |
| 1  2  3  4 | mongos> sh.enableBalancing("school.students")  WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })  mongos> sh.isBalancerRunning()  true |

Let’s check how documents get routed as per the tags:

We have inserted 6 documents, 4 documents with “subject”:”maths” and 2 documents with “subject”:”biology”

|  |  |
| --- | --- |
| 1  2  3  4  5 | mongos> db.students.find({"subject":"maths"}).count()  4  mongos> db.students.find({"subject":"biology"}).count()  2 |

Checking the shard distribution for the students collection:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | mongos> db.students.getShardDistribution()    Shard shard0000 at 127.0.0.1:27003  data : 236B docs : 4 chunks : 4  estimated data per chunk : 59B  estimated docs per chunk : 1    Shard shard0001 at 127.0.0.1:27004  data : 122B docs : 2 chunks : 1  estimated data per chunk : 122B  estimated docs per chunk : 2 |

So in this test case, all the queries for the students collection have routed as per the tag used, with four documents inserted into shard0000 and two documents inserted to shard0001.

Any queries related to MATHS will route to shard0000 and queries related to BIOLOGY will route to shard0001, hence the load will be distributed as per the configuration of the shard, keeping the database performance optimized.

Sharding MongoDB using zones is a great feature provided by MongoDB. With the help of zones, data can be isolated to the specific shards. Or if we have any kind of hardware or configuration restrictions to the shards, it is a possible solution for routing the operations.