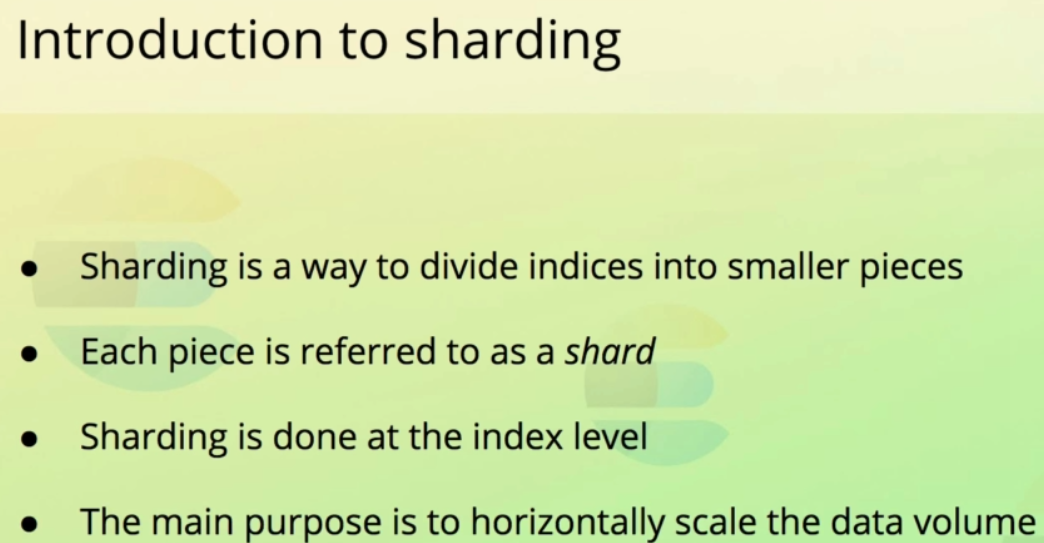
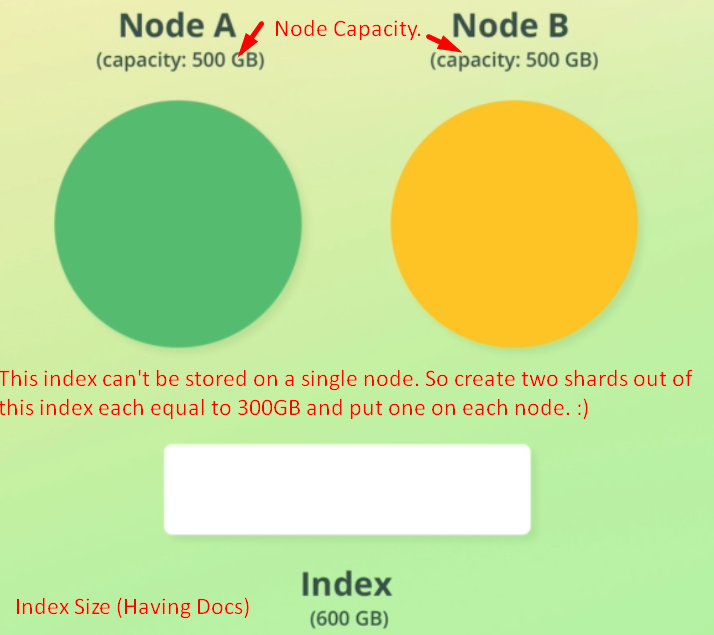
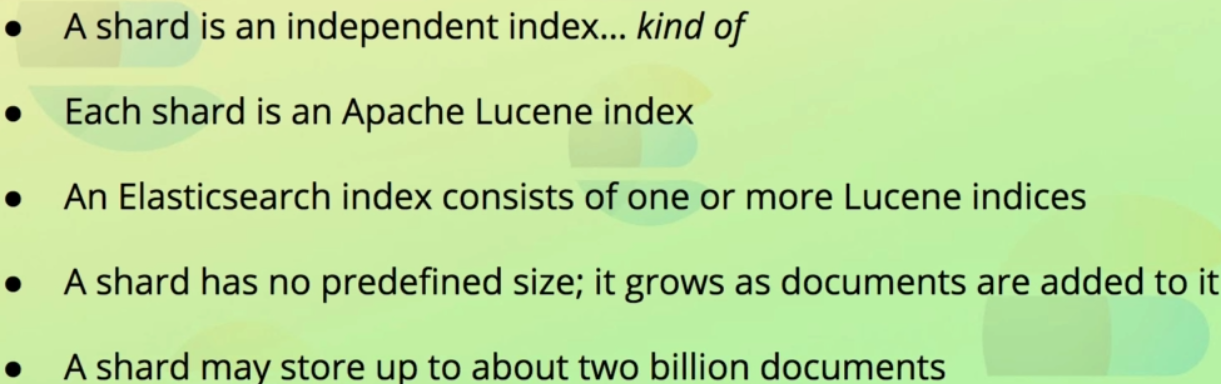
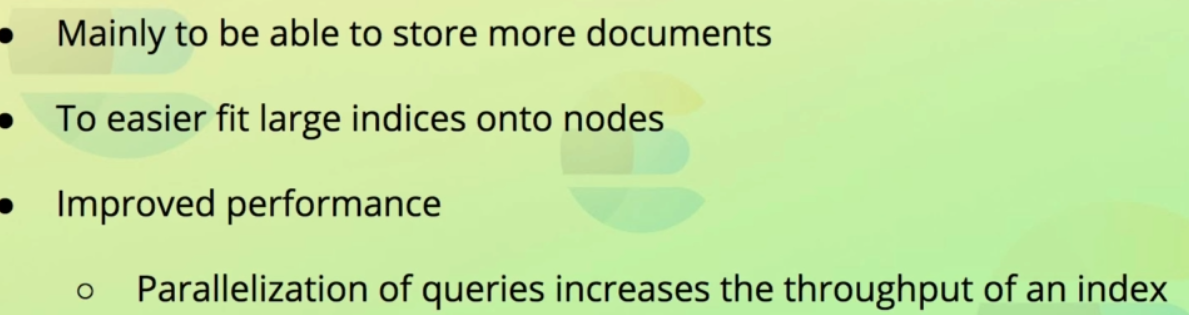
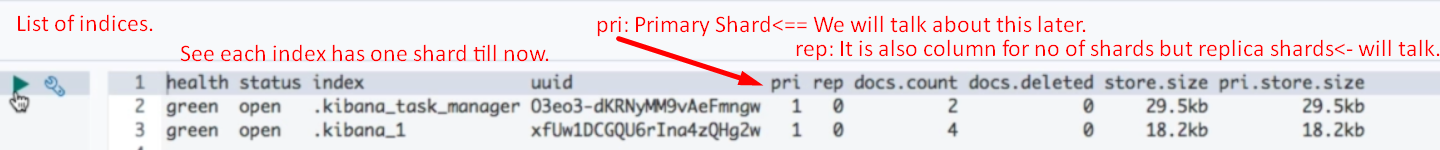
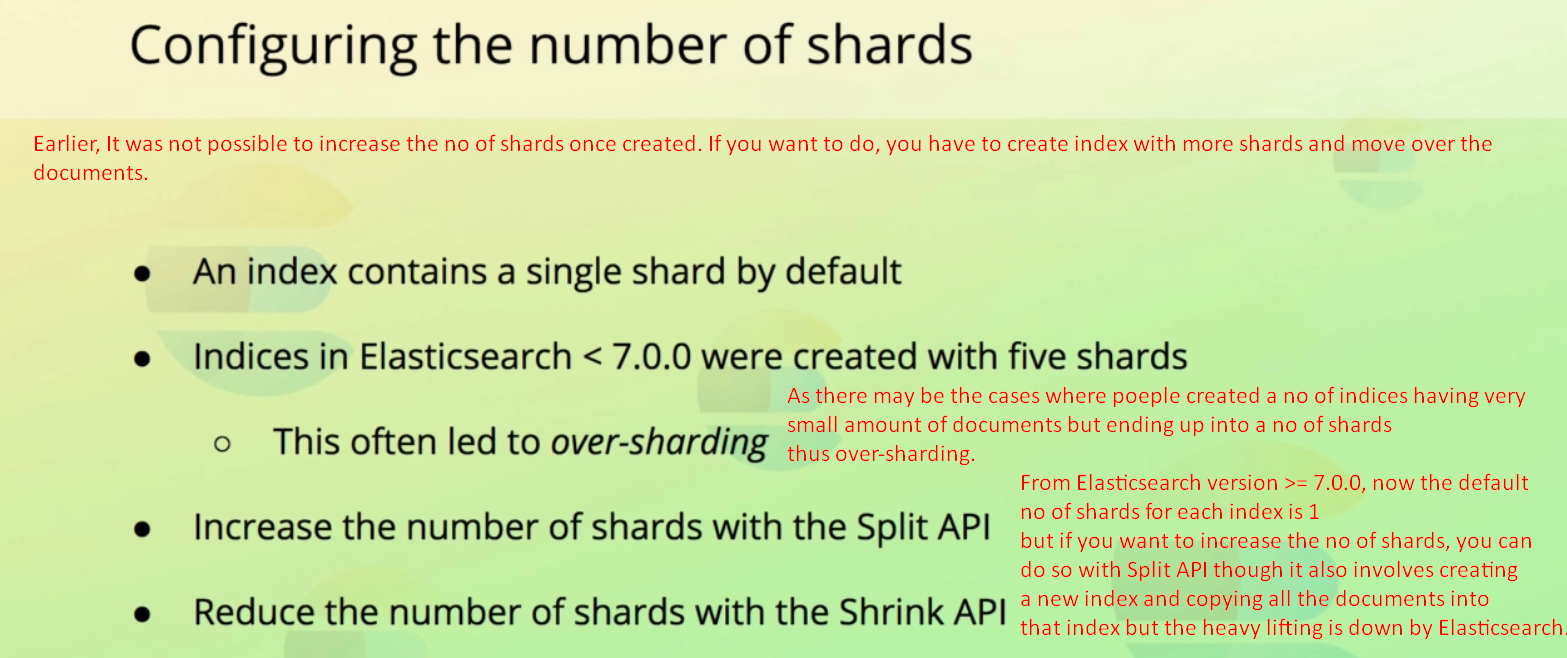
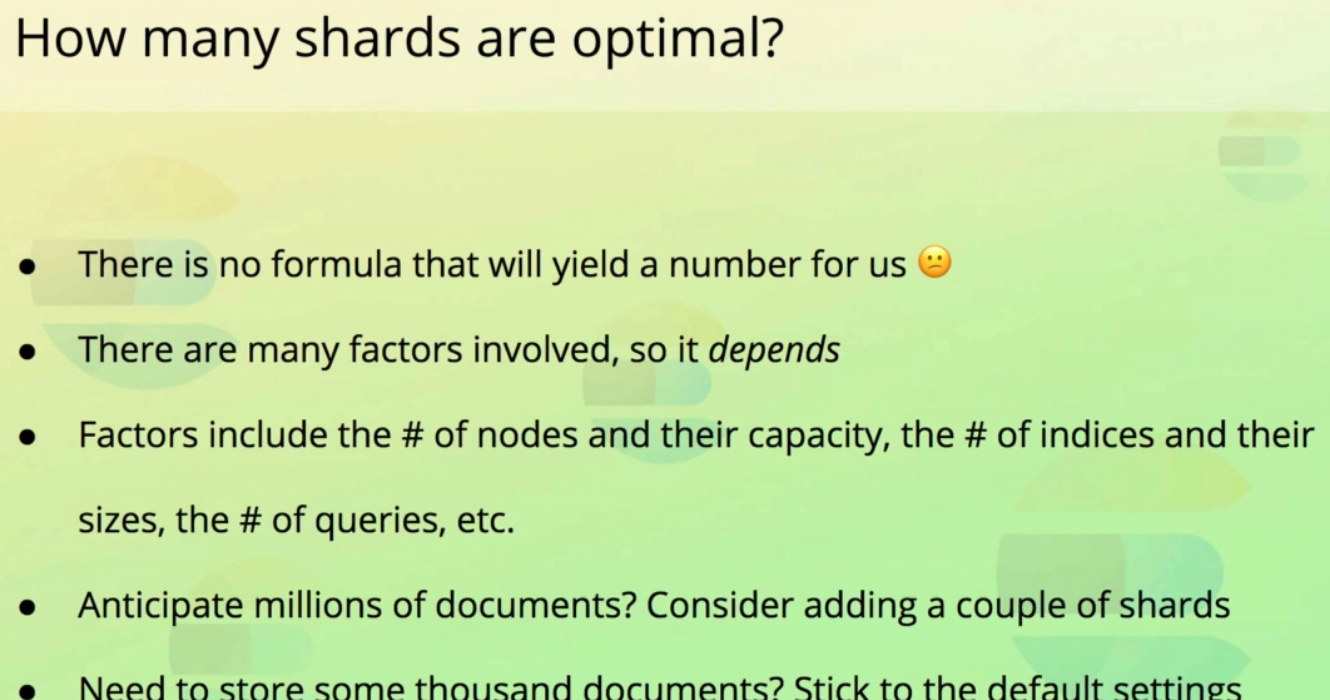
  
**Jatin**: Before start this lecture, keep in mind that index is like table in RDBMS.

1. A bit earlier, you learnt how elasticsearch consists of one or more nodes. This is one of the ways in which Elasticsearch **can scale** both in regards to data storage and disk space.
2. If we have 1TB of data and one node with 500 GB capacity that will not work. However if we add another node with at least 500GB capacity, Elasticsearch can then store data on both nodes meaning that now cluster has enough space capacity.
3. **But how does that work**? 🡺 Storing 1TB data on two nodes equal to at least capacity 1TB.
   1. Elasticsearch does this by using something **sharding.**
4. **Sharding:  
     
   Sharding** is a way to divide an index into different pieces where each piece is called shard.
   1. **Sharding** is done at index level not at node or cluster level. As you might imagine, this affects how the indices are configured.
   2. The main reason to divide index into shards is to be able to **horizontally scale** the data volume.
5. **Let’s see how sharding work**? 
   1. Suppose we have two nodes each having 500GB of storage space available for Elasticsearch.  
      **Jatin**: Nodes may be on different machines.

We have a huge index taking up to 600GB of storage on its own.

* 1. See, the whole index will not fit on either node. So running the index on a single shard is not an option because then that single shard needs to be placed on a single node.
  2. **Solution**: We divide the index into two shards each requiring 300GB worth of disk space.   
     Doing this, we can now store one shard on one node and another on another node thus without running out of disk space.
  3. We could have higher number of shards, if wanted to, such as 4 shards of 150GB each.
  4. A shard can be placed on any node. If we have space on a node and we have two shards from one index, we can put those two shards on that one node.
  5. This is how shard makes it possible to **scale the amount** of documents we can store.
  6. Each shard is **independent** and we can think of a shard as being a fully functional index on its own.  
     **NOTE**: This is not 100% accurate but closes enough.
  7. Let’s dive a bit deeper...  
     
     1. Remember how Elasticsearch is built on top of **Apache Lucene.**
     2. **Each** shard is actually a **Lucene Index**.
     3. While a shard doesn’t have a **predefined size** in terms of allocated disk space, there is a limit to the number of documents a shard can store, being just over two billion documents.

1. **The purpose of sharding**:  
   
   1. The main reason for sharding an index is to able to **scale its data volume**, being the number of documents it can store.  
      By using sharding, you can store billions of documents within a single index which would not be feasible without sharding.
   2. It enables queries to be **distributed** and **parallelized** across an index’ shards as different shards may be on different machines (true even if on same machine). It means that search query can be run on multiple shards at the same time thus increasing the performance and throughput.
2. 
3. 
4. **Pri column**: 
5. **NOTE**: By default, each index is stored on a single shard. But this can be configured when creating index.
6. 
7. 
8. 