Introduction to sharding

- Sharding is a way to divide indices into smaller pieces
- Each piece is referred to as a *shard*
- Sharding is done at the index level
- The main purpose is to horizontally scale the data volume

Let’s dive a bit deeper...

- A shard is an independent index... *kind of*
- Each shard is an Apache Lucene index
- An Elasticsearch index consists of one or more Lucene indices
- A shard has no predefined size; it grows as documents are added to it
- A shard may store up to about two billion documents

The purpose of sharding

- Mainly to be able to store more documents
- To easier fit large indices onto nodes
- Improved performance
    - Parallelization of queries increases the throughput of an index

Configuring the number of shards

- An index contains a single shard by default
- Indices in Elasticsearch < 7.0.0 were created with five shards
    - This often led to *over-sharding*
- Increase the number of shards with the Split API
- Reduce the number of shards with the Shrink API

How many shards are optimal?

- There is no formula that will yield a number for us 😕
- There are many factors involved, so it *depends*
- Factors include the # of nodes and their capacity, the # of indices and their sizes, the # of queries, etc.
- Anticipate millions of documents? Consider adding a couple of shards
- Need to store some thousand documents? Stick to the default settings

Lecture summary

- Sharding splits indices into smaller pieces
- Sharding increases the number of documents an index can store
- Sharding makes it easier to fit large indices onto nodes
- Sharding may improve query throughput
- An index defaults to having *one* shard
- Add a couple of shards for large indices; otherwise use default settings