ElasticSearch tutorial

What is a *"Node"*?

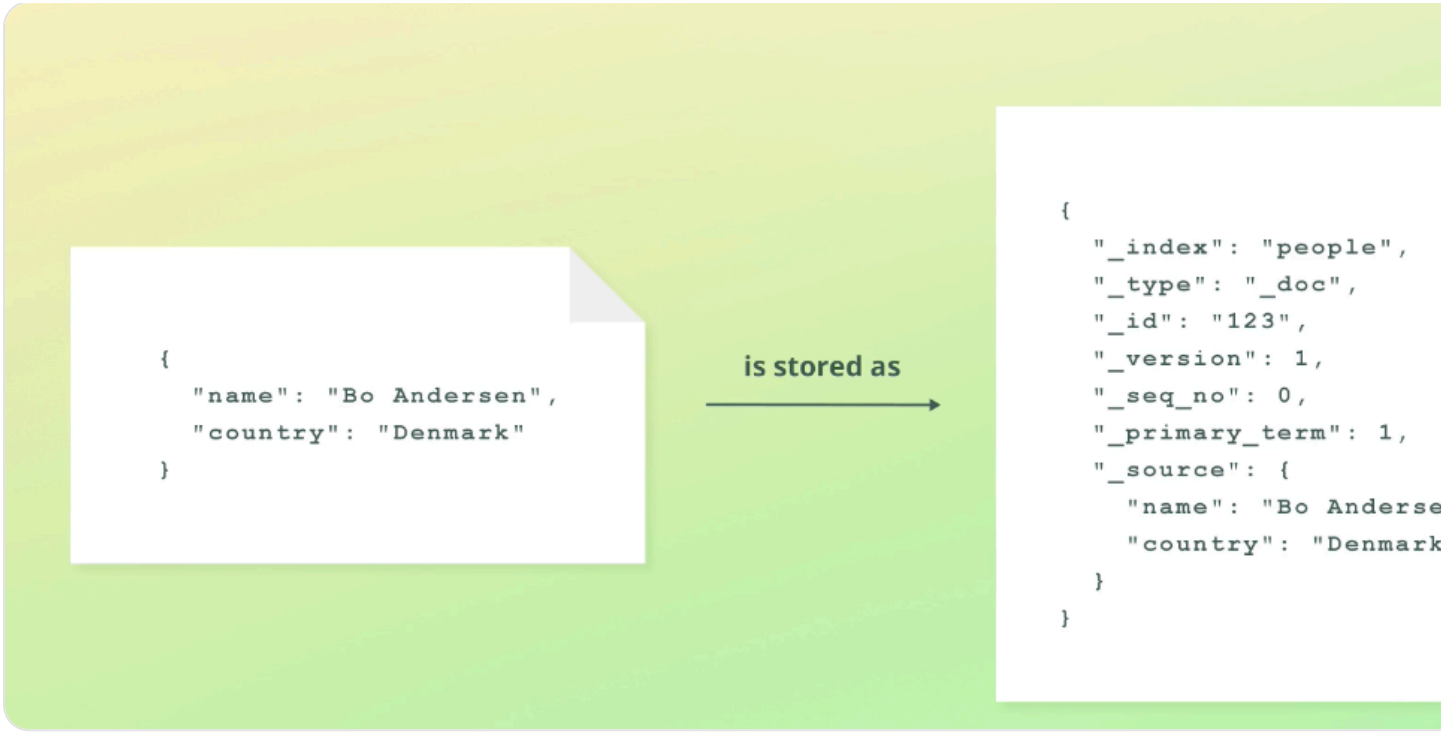
- node is an instance of ElasticSearch that stores data.
- To ensure that we can store many terabytes of data if we need to, we can run as many
  - Each node will then store a part of our data. aka (Sharding).
- Node refers as an instance of Elasticsearch and not a machine.
  - so you can run any number of nodes on the same machine.
- That being said, you should typically separate things in a production environment so that each node runs on a dedicated machine, a virtual machine, or within a container.
- each node belongs to what is called a Cluster.

What is a *"Cluster"*?

- A cluster is a collection of related nodes that together contain all of our data.
- We can have many clusters if we want to, but one is usually enough.
- It is possible to perform cross-cluster searches, but it is not very common to do so.
- You might run multiple clusters that serve different purposes; for instance, you could
  - have a cluster for powering the search of an e-commerce application,
  - and another for Application Performance Management (abbreviated APM).
- The reasons for splitting things into multiple clusters, are typically to separate things logically, and to be able to configure things differently.

What is a *"Document"*?

- Each unit of data that you store within your cluster is called a document.
- Documents are JSON objects containing whatever data you desire.
- When you index a document, the original JSON object that you sent to Elasticsearch is stored
- along with some metadata that Elasticsearch uses internally.



What is an *"Index"*?

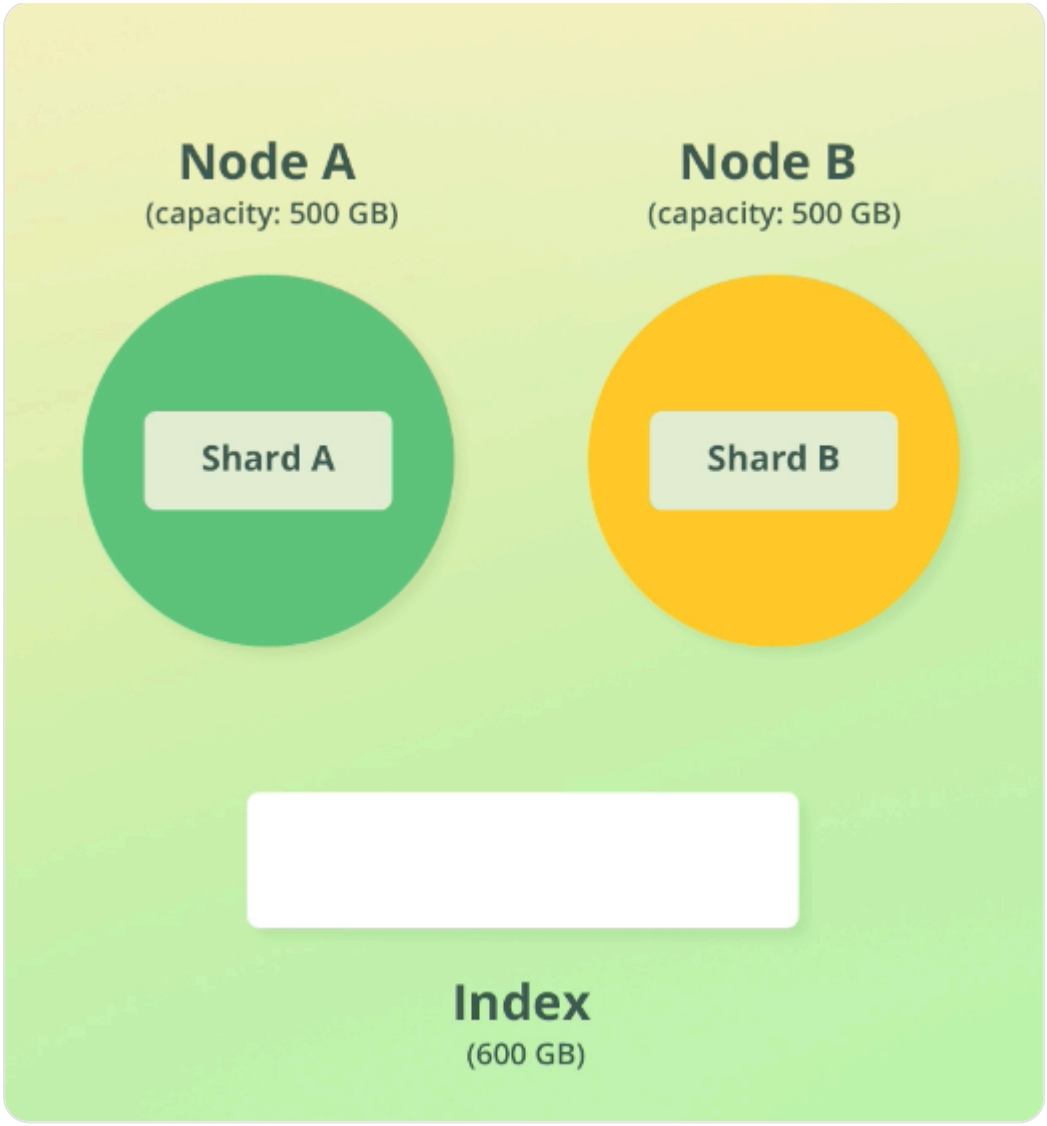
- So how are documents organised, you might wonder?
  - The answer is within indices.
- An index groups documents together logically, as well as provide configuration options that
  - are related to scalability and availability, which we will take a look at a bit later.
- When we get to searching for data, you will see that we specify the index that we want
  - to search for documents, meaning that search queries are actually run against indices.
  - meaning we need index to search for data.

What is a *"Shard"* ?

- 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.

Example of when sharding is needed?

- index = `600 GB`
- 2 nodes each one has capacity = `500 GB`.
- Therefore, running the index on a single shard is not an option, because a shard needs to be placed on a single node.
  - Solution-1: Instead, we can divide the index into two shards, each requiring `300 gigabytes` worth of disk space.
  - Solution-2: We could also have a higher number of shards if we wanted to, such as four shards of `150 gigabytes` each.
- We still have space to spare, so we could use that for other indices if we needed to.
- Just to be clear, a shard may be placed on any `node`:
  - so if an `index` has `5 shards`, for example, we `DO NOT` need to spread these out on `5 different nodes`.



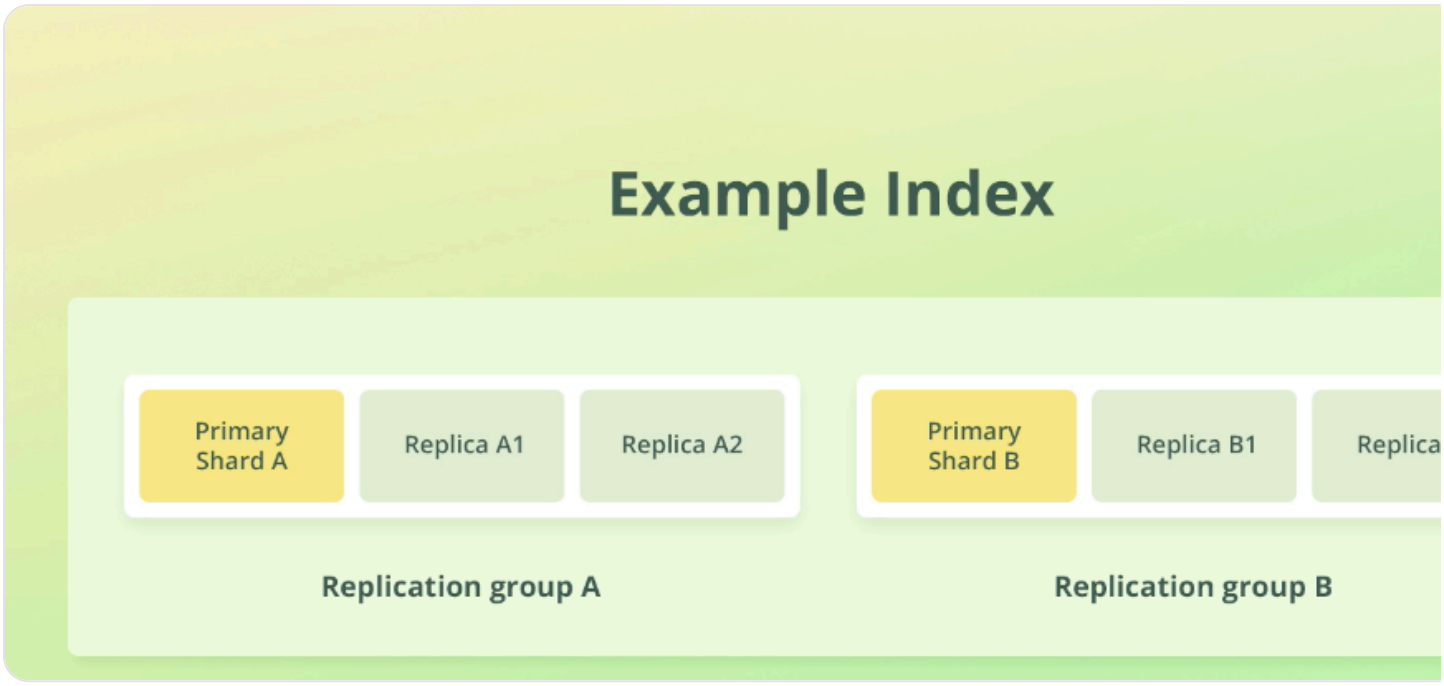
- `pri` is short of ***"Primary Shard"***

	health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
2	green	open	.internal.alerts-transform.health.alerts-default-000001	ug43hgUnQS6n9zAI1cHqXw	1	1	0	0	498b	249b
3	green	open	.ds-metrics-fleet_server.agent_status-default-2024.04.27-000001	QrsnAxFvTzKXOBUpklITDQ	1	1	137	0	309.4kb	154.1kb
4	green	open	.internal.alerts-observability.logs.alerts-default-000001	RmIG9GwPQsivaNccrTBMpg	1	1	0	0	498b	249b
5	green	open	.internal.alerts-observability.uptime.alerts-default-000001	OI5yQVy2SM20uQEms6yxiQ	1	1	0	0	498b	249b
6	green	open	.internal.alerts-ml.anomaly-detection.alerts-default-000001	FORheHlDSee-MTB0TTS50Q	1	1	0	0	498b	249b
7	green	open	.ds-logs-enterprise_search.api-default-2024.04.27-000001	2FOLyEFLQOe2XsLQgGXGaQ	1	1	2	0	63.8kb	31.9kb
8	green	open	.internal.alerts-observability.slo.alerts-default-000001	ag5Ol-BlROyJ-v-Hl6KZpQ	1	1	0	0	498b	249b
9	green	open	.internal.alerts-default.alerts-default-000001	VqYO-TL0RVmhGU7zKMejyw	1	1	0	0	498b	249b
10	green	open	.internal.alerts-observability.apm.alerts-default-000001	ml7YMa0ASUGXhik9DutO2g	1	1	0	0	498b	249b
11	green	open	.ds-logs-enterprise_search.audit-default-2024.04.27-000001	XI2yvcI_QKCozXg1T-qq0Q	1	1	8	0	129.7kb	64.8kb
12	green	open	.elastic-connectors-v1	E9Vvf_KXQrmI4AYwUL3acQ	1	1	0	0	7.6kb	247b
13	green	open	.internal.alerts-observability.metrics.alerts-default-000001	pTnSZXtnTZi5R1bA2FRYwg	1	1	0	0	498b	249b
14	green	open	.kibana-observability-ai-assistant-conversations-000001	NF-9kGbNTxeBl71zIha8YQ	1	1	0	0	498b	249b
15	green	open	.internal.alerts-ml.anomaly-detection-health.alerts-default-000001	T1SfSt_nQTWi4fFzogxayw	1	1	0	0	498b	249b
16	green	open	.internal.alerts-observability.threshold.alerts-default-000001	V4y1fi0PRsKsDMrbzywgqQ	1	1	0	0	498b	249b
17	green	open	.elastic-connectors-sync-jobs-v1	EsKmh__dTLiRO2UDVfpyfQ	1	1	0	0	7.6kb	247b
18	green	open	.internal.alerts-security.alerts-default-000001	Cd7KH2VST1GDheeN40dJRw	1	1	0	0	498b	249b
19	green	open	.kibana-observability-ai-assistant-kb-000001	s2S7PpZhQMumeG-vQ-ucEw	1	1	0	0	498b	249b
20	green	open	.ds-metrics-fleet_server.agent_versions-default-2024.04.27-000001	Gc2FFH0qTeahfc8-emC1zw	1	1	137	0	228.2kb	113.5kb
21	green	open	.internal.alerts-stack.alerts-default-000001	dnsVdBL7T5u5c8eAQ50x9g	1	1	0	0	498b	249b

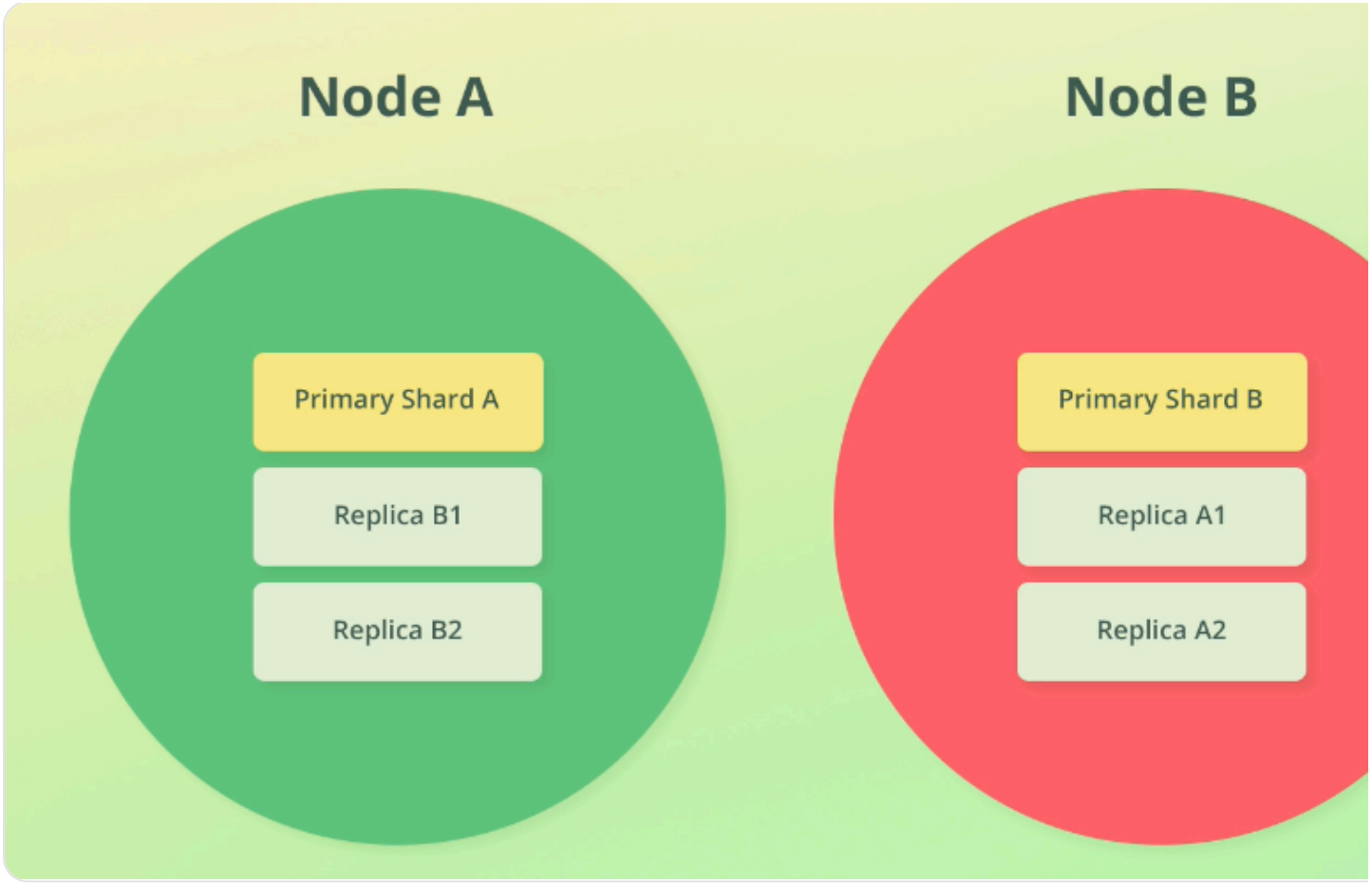
What is a "*Replica Shard*"?

- Elasticsearch supports replication for fault tolerance.
- Replication is supported natively and enabled by default.
- Replication is configured at the index level.
- Replication works by creating copies of shards, referred to as ***"Replica Shards"***.
- A shard that has been replicated, is called a ***"Primary Shard"***: `pri` .

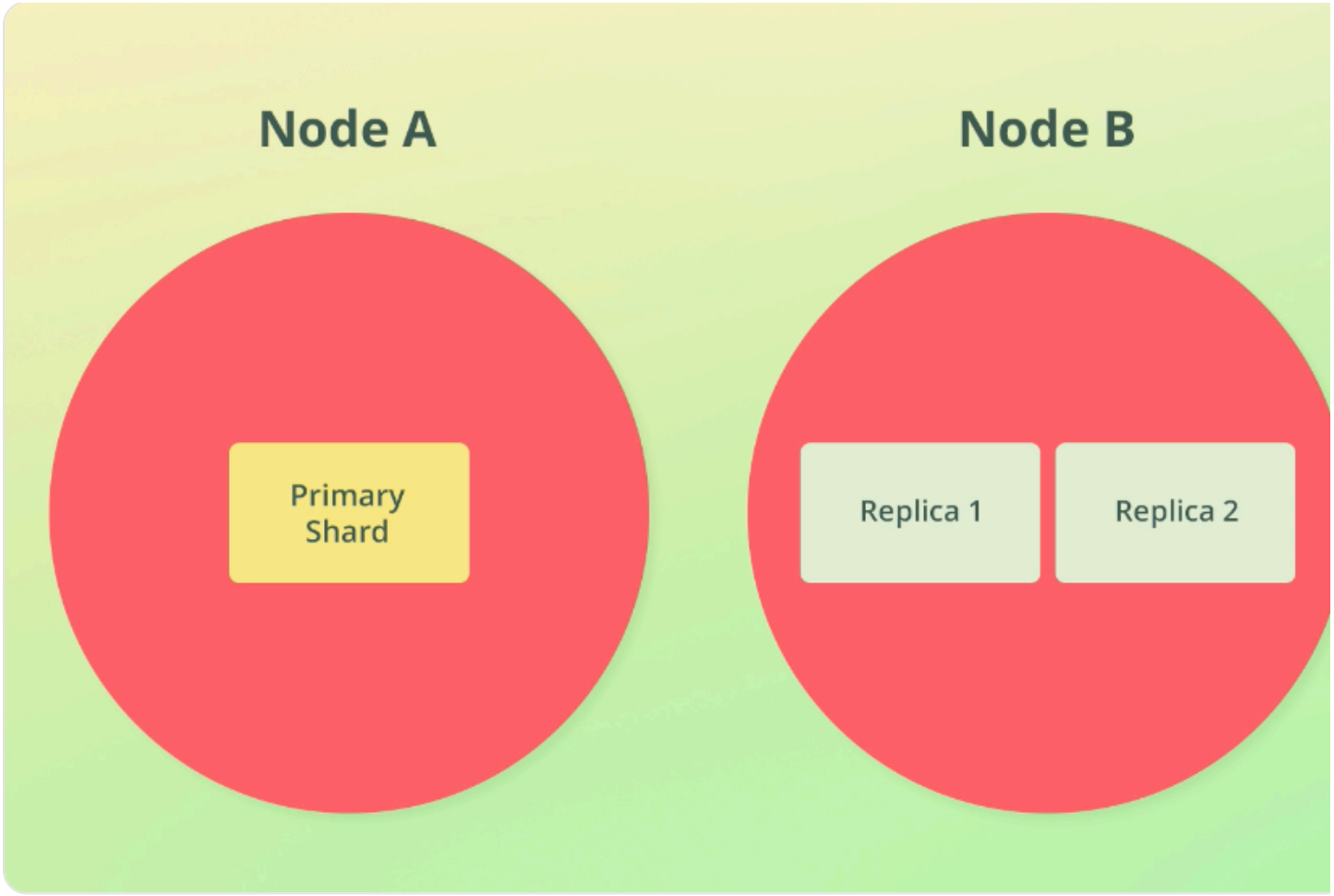
- A `primary shard` and its `replica shards` are referred to as a `replication group`
- `replication group` is used in search queries.



- A `replica shard` is ***NEVER*** stored on the same `node` as its `primary shard`.
- For this reason, Elasticsearch will ***ONLY ADD*** `replica shards` for clusters with `multiple nodes`.
- Typically you would be fine with `one or two` replicas, but that depends on how critical your setup is.
  - As a rule of thumb 👍:
    - you should replicate shards `once` for normal usage.
    - and for `critical systems`, you should replicate them `twice or more`.

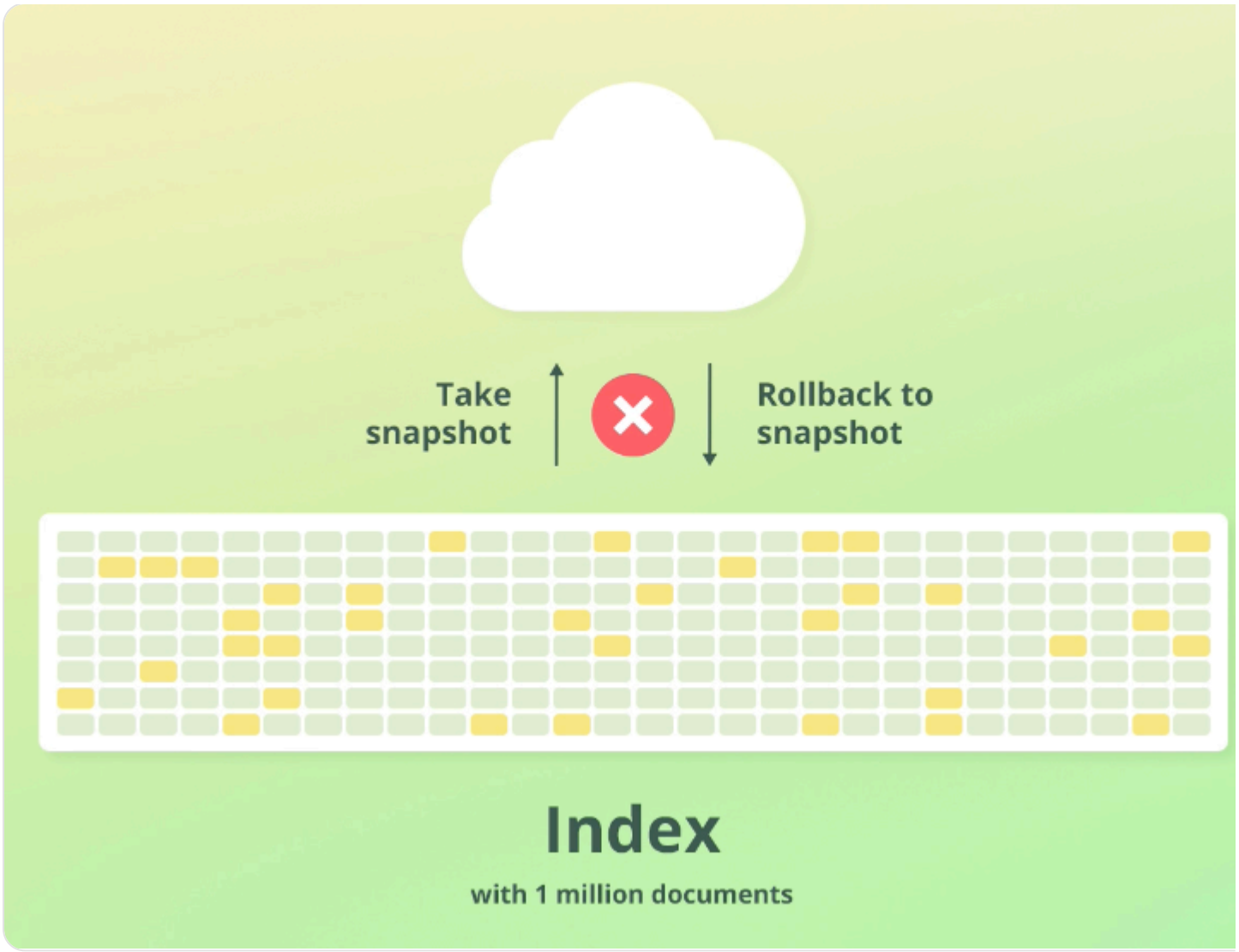


- `Replication` just ensures that indices can recover from a `node failure` (hardware failure) and keep serving requests as if nothing bad happened.
- Apart from preventing data loss, `Replication` can be used to increase the throughput (it helps with search READ queries) of a given index.
  - in this example you don't need another Shard as you don't have that many data. However, your application requires a lot of reads.
    - Note: having shard-replicas on the same node will only be helpful if the hardware resources of the node have not yet been fully utilised.
    - If the nodes were already busy handling requests for other indices, we would see little to no effect of adding an additional replica shard.
    - it also requires desk space as we replicated the entire `pri`



What is a *“Snapshots”*?

- Snapshots are commonly used for daily backups and manual snapshots may be taken before applying
- changes to data, just to make sure that there is a way to roll back the changes in case something goes wrong.
- Replication cannot help with rolling back, because replication just ensures that we don't lose our latest data, which has already been modified in this example.


Take
snapshot
Rollback to
snapshot
Index
with 1 million documents