elastic

[639] Search and indexing

Meenakshi Syamkumar

Learning Objectives

Recognize	use cases for Elasticsearch
Define	key features of Elasticsearch
Describe	deployment options and terminologies
Identify	differences between MongoDB & Elasticsearch in the context of text search
Define	indexing & shards

What is Elasticsearch?

- Near real-time, distributed, open-source, RESTful, highly scalable search and analytics engine
  - horizontal scalability
- Elastic Stack (formerly ELK Stack)
  - Beats: data ingestion aka collection
  - Logstash: data aggregation & monitoring
  - Elasticsearch: search & indexing
  - Kibana: visualization
- Works with different types of data: text, numeric, geospatial, structured, semi-structured, and unstructured

Use cases for Elasticsearch

- Full-text, application, website & enterprise search
- Log and event analytics
- Infrastructure metrics & container monitoring
- Application performance monitoring
- Geospatial data analysis (& visualization using Kibana)
- Security & business analytics

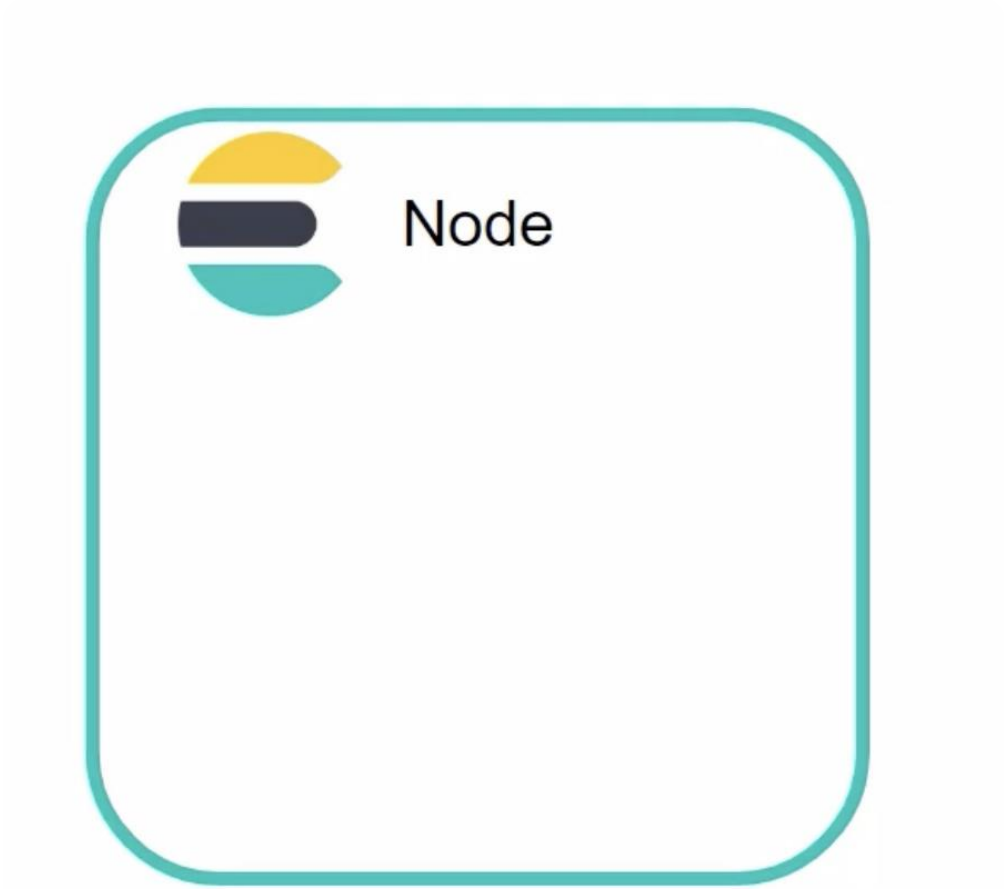
Key features of Elasticsearch

- Distributed architecture
  - data is partitioned into **shards** and distributed across the cluster
- RESTful API
  - JSON-based
  - interaction with HTTP requests
- Full-text search
  - underlying engine: Apache Lucene
  - text analysis, relevance ranking fuzzy searches, etc.,
- Near real-time search
  - applications where up-to-date information is critical

Java™

Key features of Elasticsearch

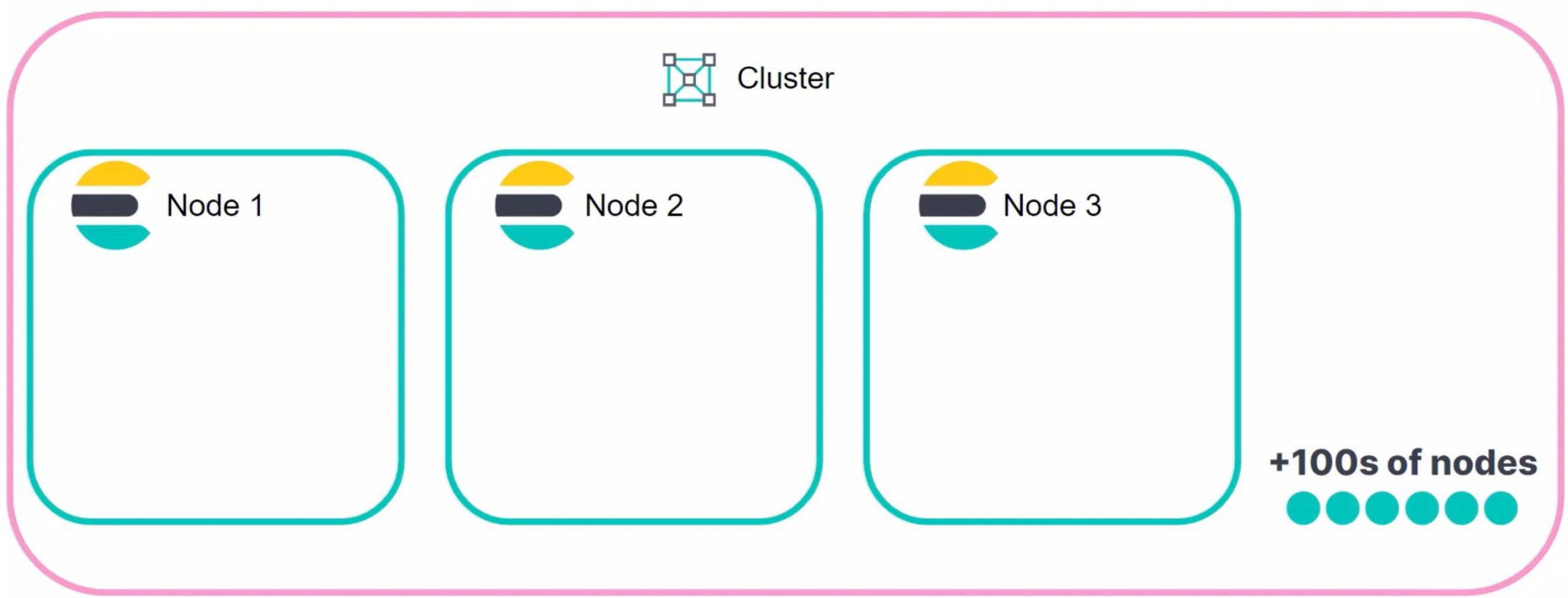
- Schema-free
  - does support explicit mappings
  - dynamically index fields without predefined schema
  - ideal for semi-structured and unstructured data
- Aggregation
  - powerful analytical queries to summarize and process large datasets efficiently
- Scalability
  - petabyte-scale deployments
- Near real-time indexing
  - new documents become searchable within seconds



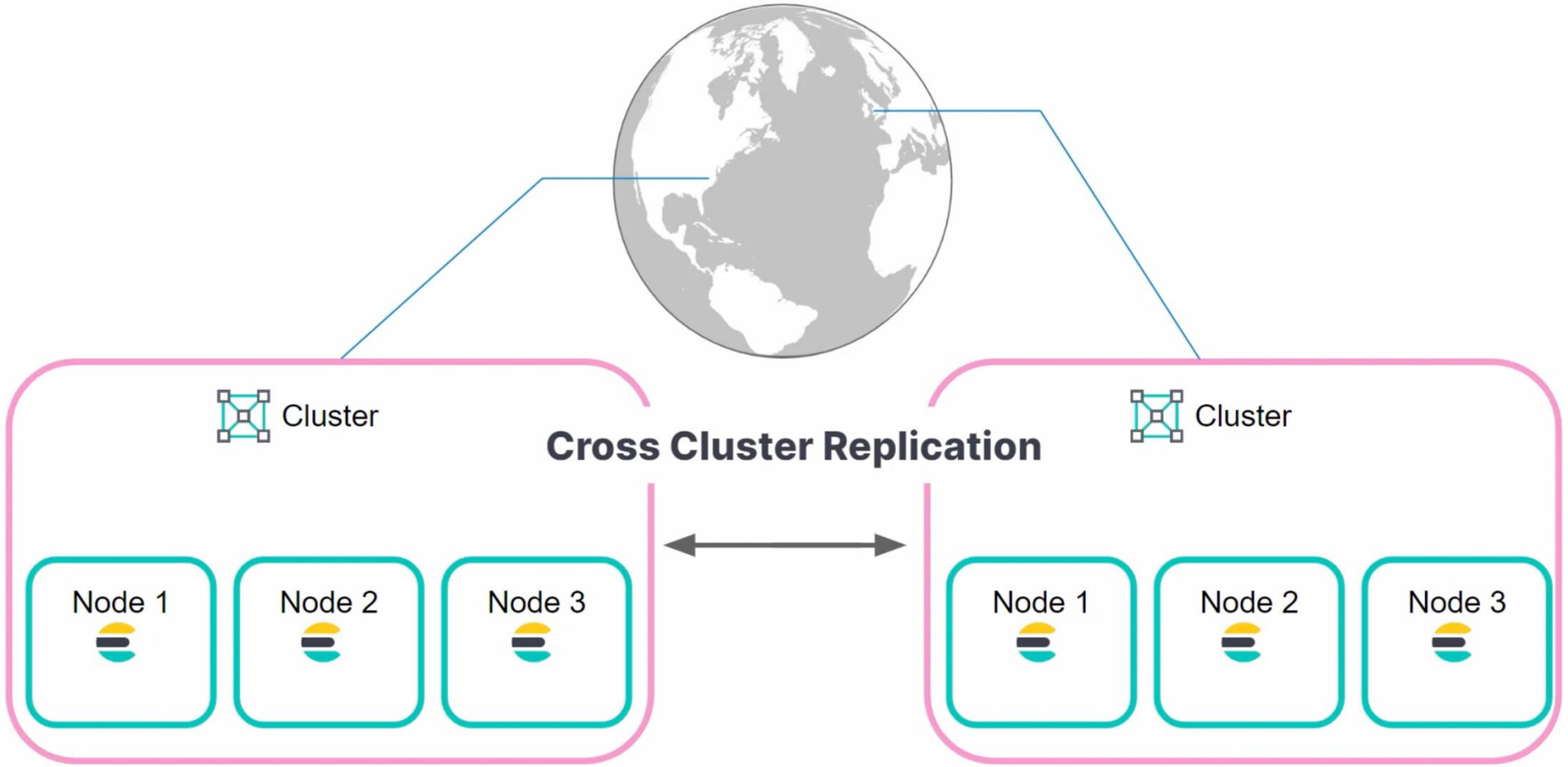
Deployment

- Node:
  - single instance of elasticsearch
  - can be installed on:
    - hardware
    - virtual machines
    - containers
    - cloud

Cluster deployment



Geographic cluster deployments



Elasticsearch: document store

- Distributed document store
- A document is a serialized JSON object that is stored in Elasticsearch under a unique document id

A JSON object ...

```
{
    "title": "You Know, for Search",
    "author_first_name": "Shay",
    "author_last_name": "Banon",
    "post_date": "2010-02-08T19...",
    "body_l10n": "Elasticsearch is an
open source, distributed, RESTful,
search engine which is built...",
    ...
}
```

- is stored in Elasticsearch as a document

Documents are JSON objects

- How can we represent unstructured data using JSON?

title	category	date	author_first_name	author_last_name	author_company	Blog Text
You Know, for Search	Engineering	February 08, 2010	Shay	Banon	Elastic	Elasticsearch is a distributed…
How we saved $100,000/month by keeping our own software up to date	User Stories	April 27, 2022	George	Kobar	Elastic	Let's start with the bottom line…

A document consists of ***fields…***

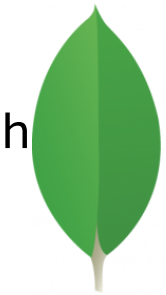
```
{
   "title": "You Know, for Search",
   "author_first_name": "Shay",
   "author_last_name": "Banon",
   "post_date": "2010-02-08T19...",
   "body_l10n": "Elasticsearch is a
distributed, RESTful, search engine which is
built...",
   ...
}
```

… and ***values***

Why Elasticsearch?

MongoDB or Elasticsearch?

- Single text index per collection
  - Index can span multiple fields
- Schema-less
  - Dynamically index documents only after index creation
- Text search ($text operator)
  - Limited to the fields defined as part of the index
  - Features:
    - Exact phrase & case-insensitive search
    - Stemming & stop words
    - Basic Boolean logic



mongoDB®

MongoDB or Elasticsearch?

- Inverted index (Apache Lucene)
  - Optimized for large-scale, high-performance searc
- Dynamic field indexing
  - No need for predefined text indices
  - All fields are automatically indexed
- Multiple index types
  - Keyword, text, etc.,

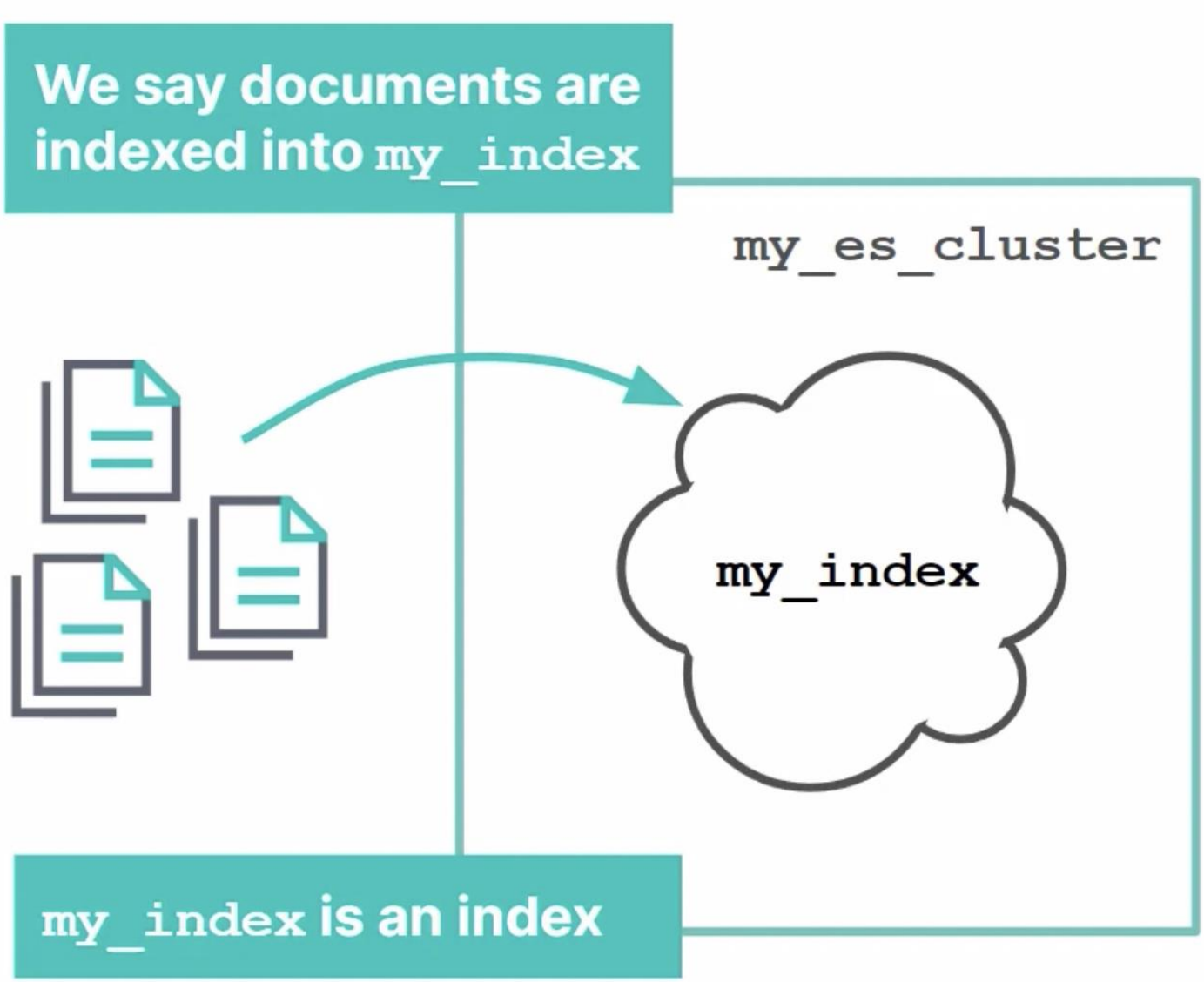
MongoDB or Elasticsearch?

- Query language (DSL)
  - Advanced full-text search:
    - Fine-tuned full-text searches, exact phrase matching, fuzzy matching, wildcard searches, proximity searches
  - Fuzzy matching: misspelling or minor variants
  - Complex Boolean logic
    - must, should, should_not
  - Scoring & boosting
    - BM25
    - Customize the ranking of search results based on fields, boost factors, and custom scoring logic
  - Geo-search and nested documents

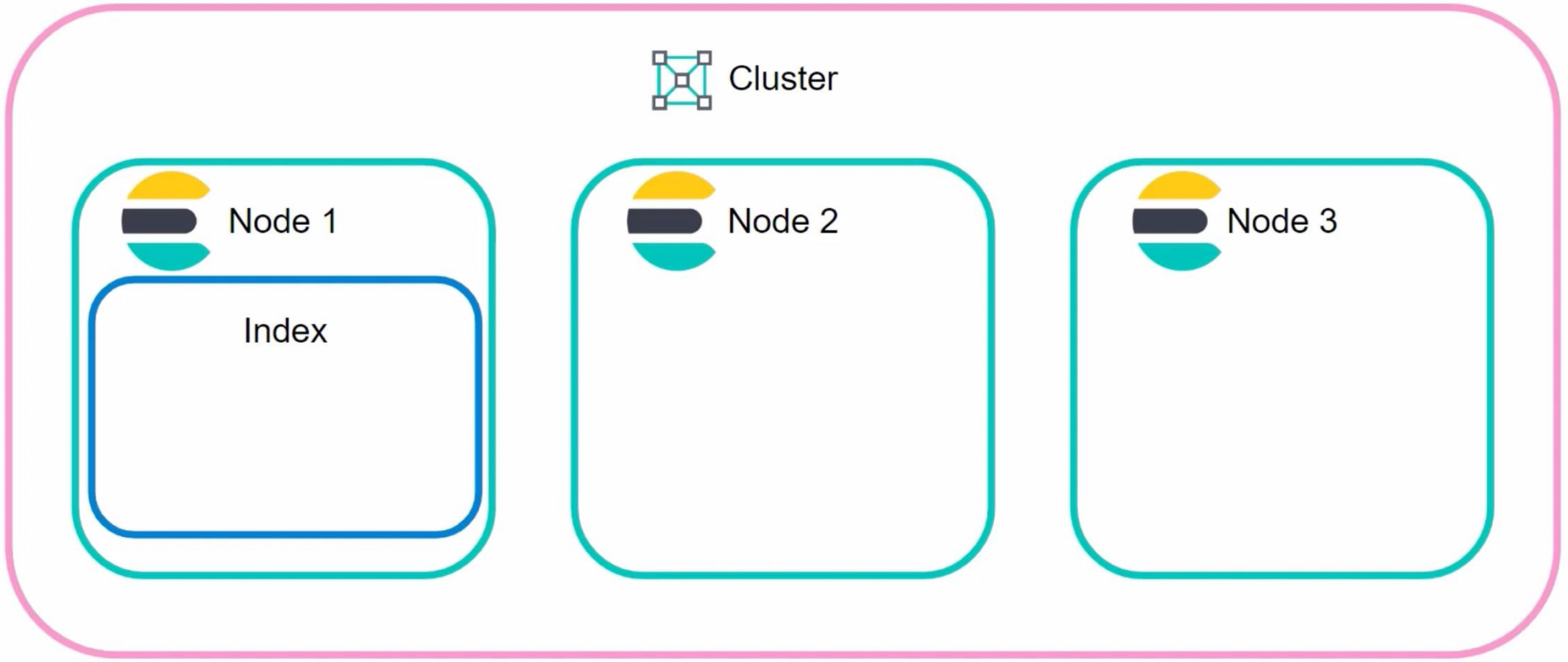
Indexing

Documents are indexed into an index

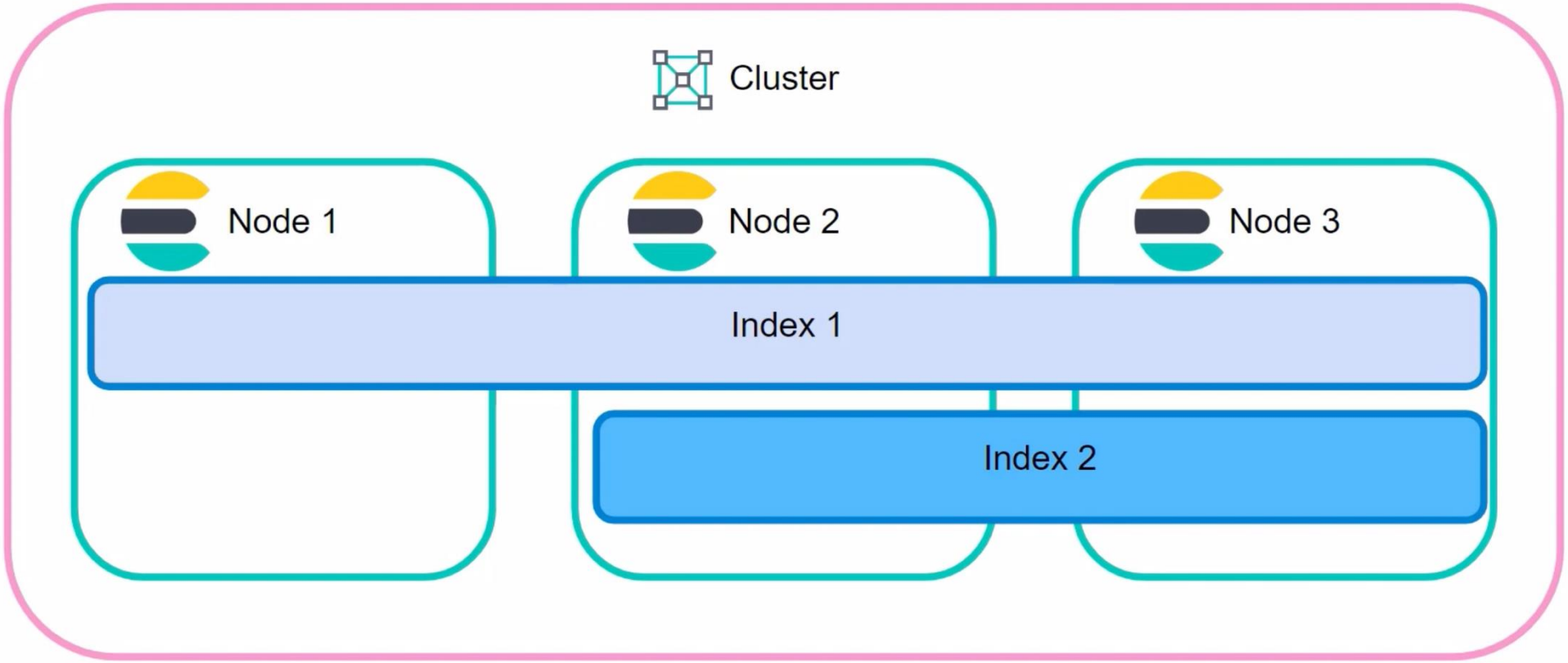
- document is indexed into an ***index***
- index
  - logical way of grouping data
  - can be thought of as an optimized collection of documents



Index is stored in an elasticsearch node

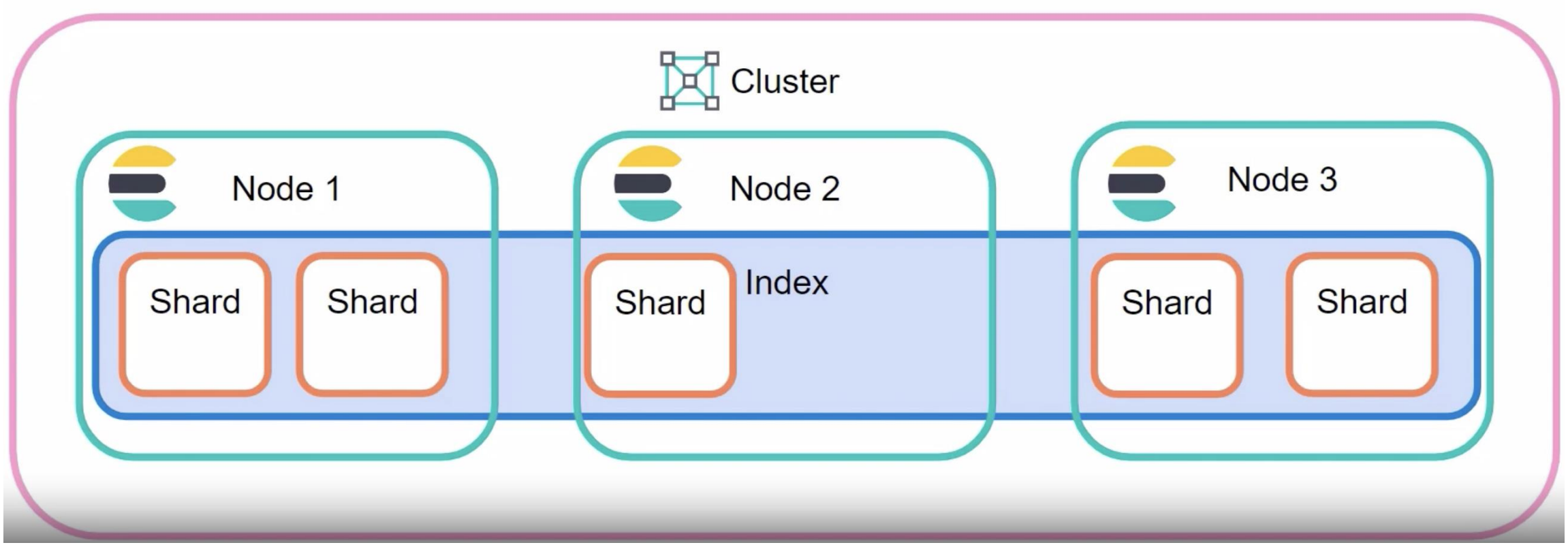


Indices can span the entire cluster

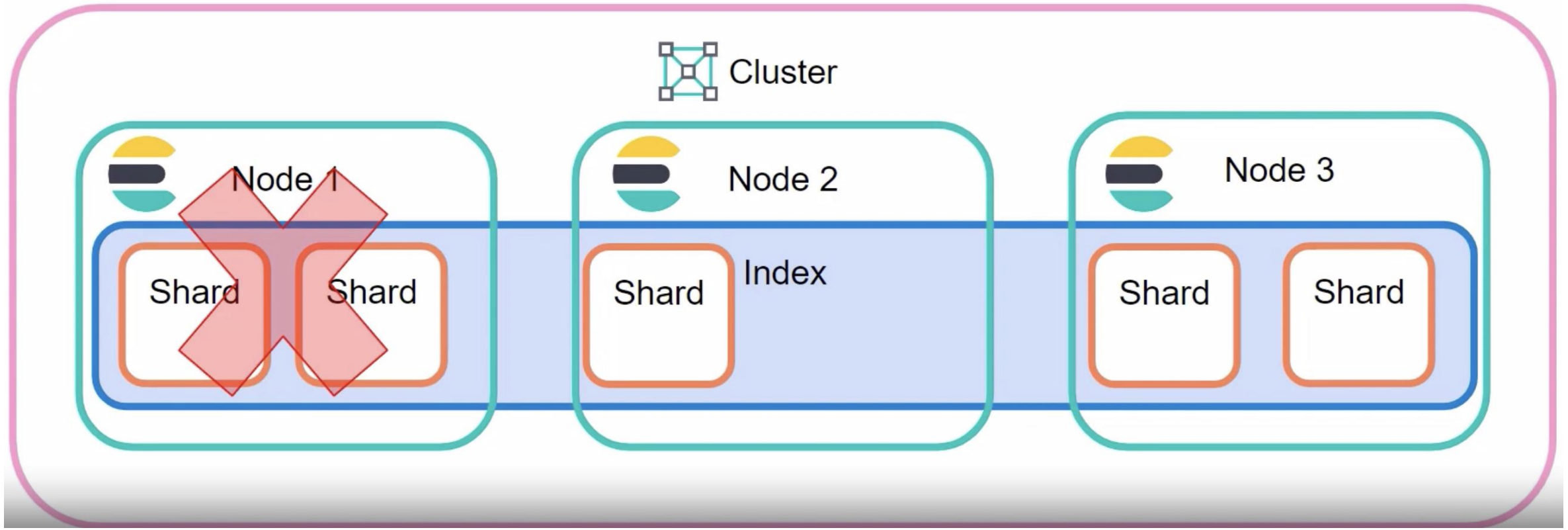


Shards

- Index distributes documents over one or more shards
- Each shard:
  - Instance of Lucene
  - Contains all the data of any one document

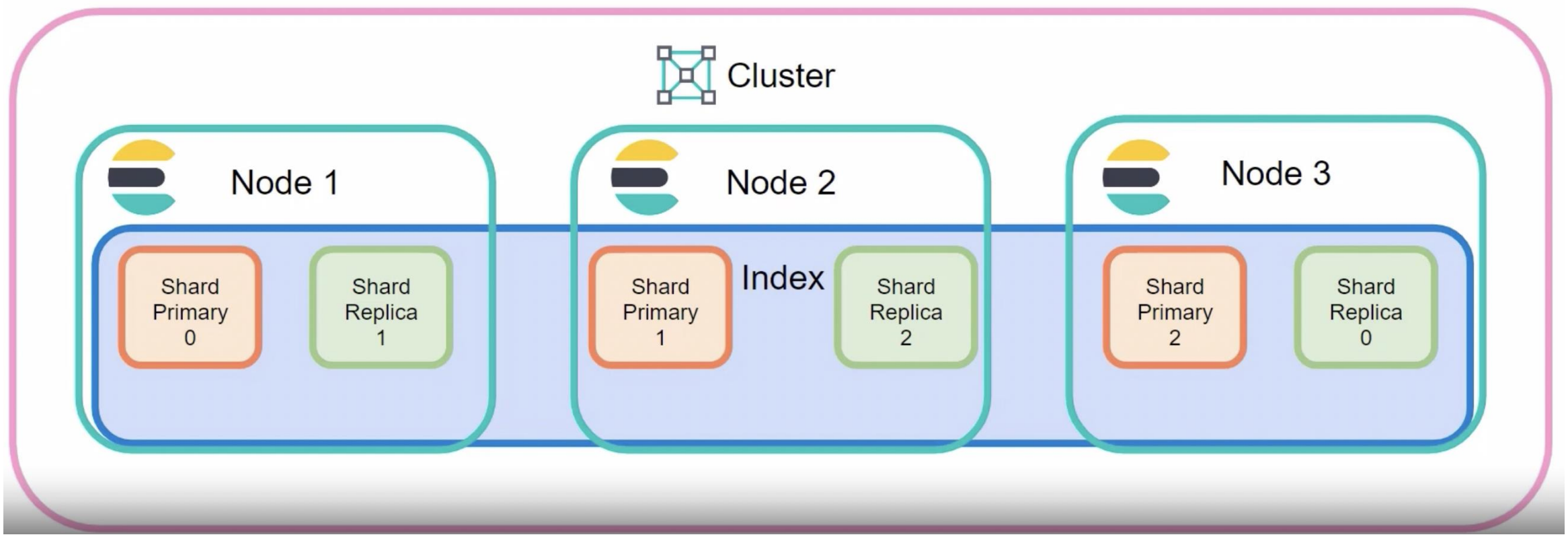


Data loss?

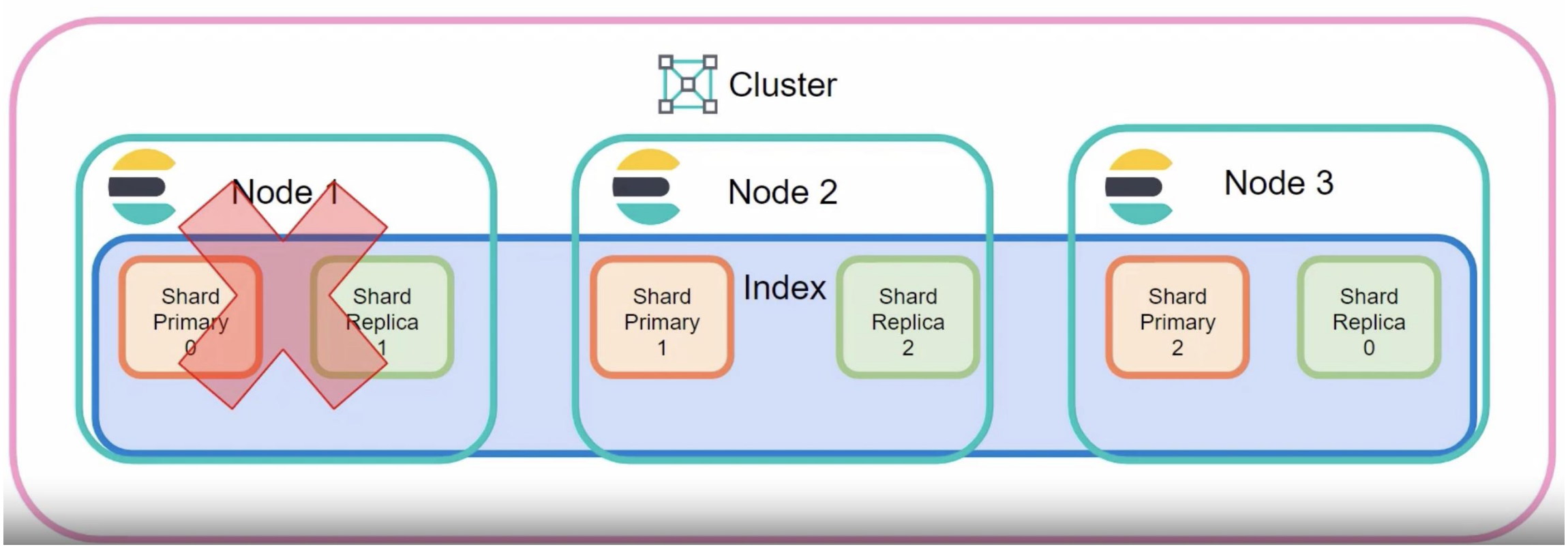

Cluster
Node 1
Shard
Shard
Node 2
Shard
Index
Node 3
Shard
Shard

Replication

- Two types of shards:
  - primary shards: the original shards of the index
  - replica shards: copies of the primary shards

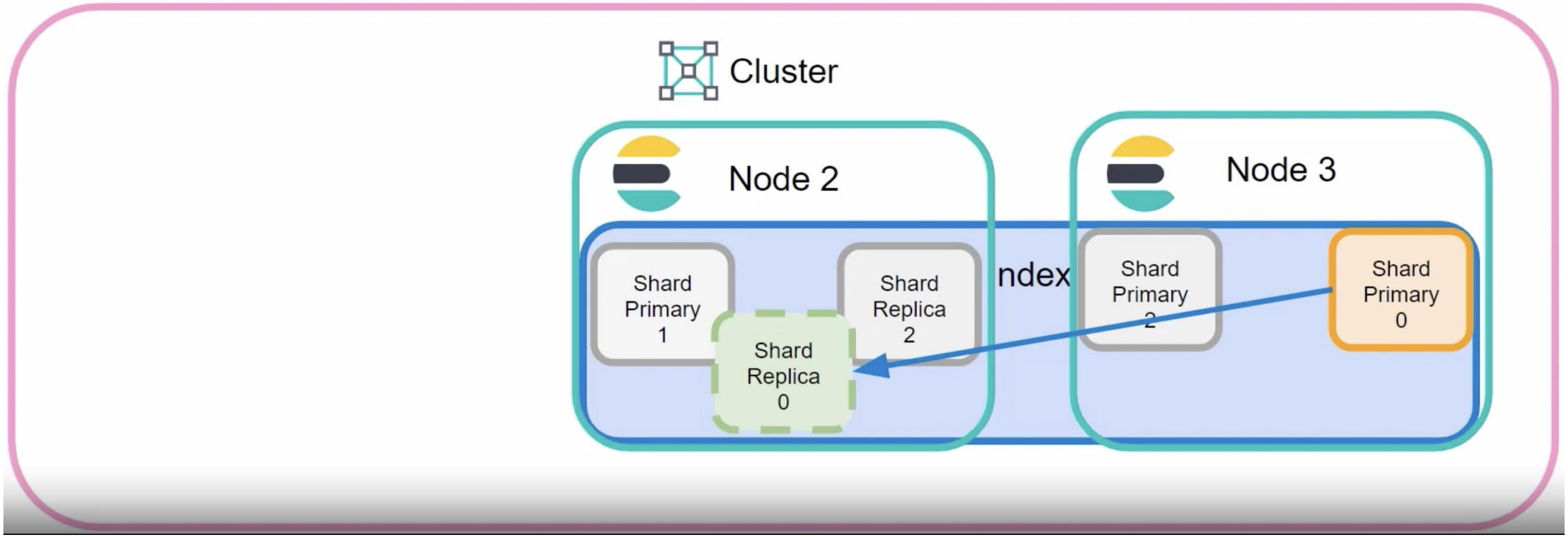


Node loss with replication

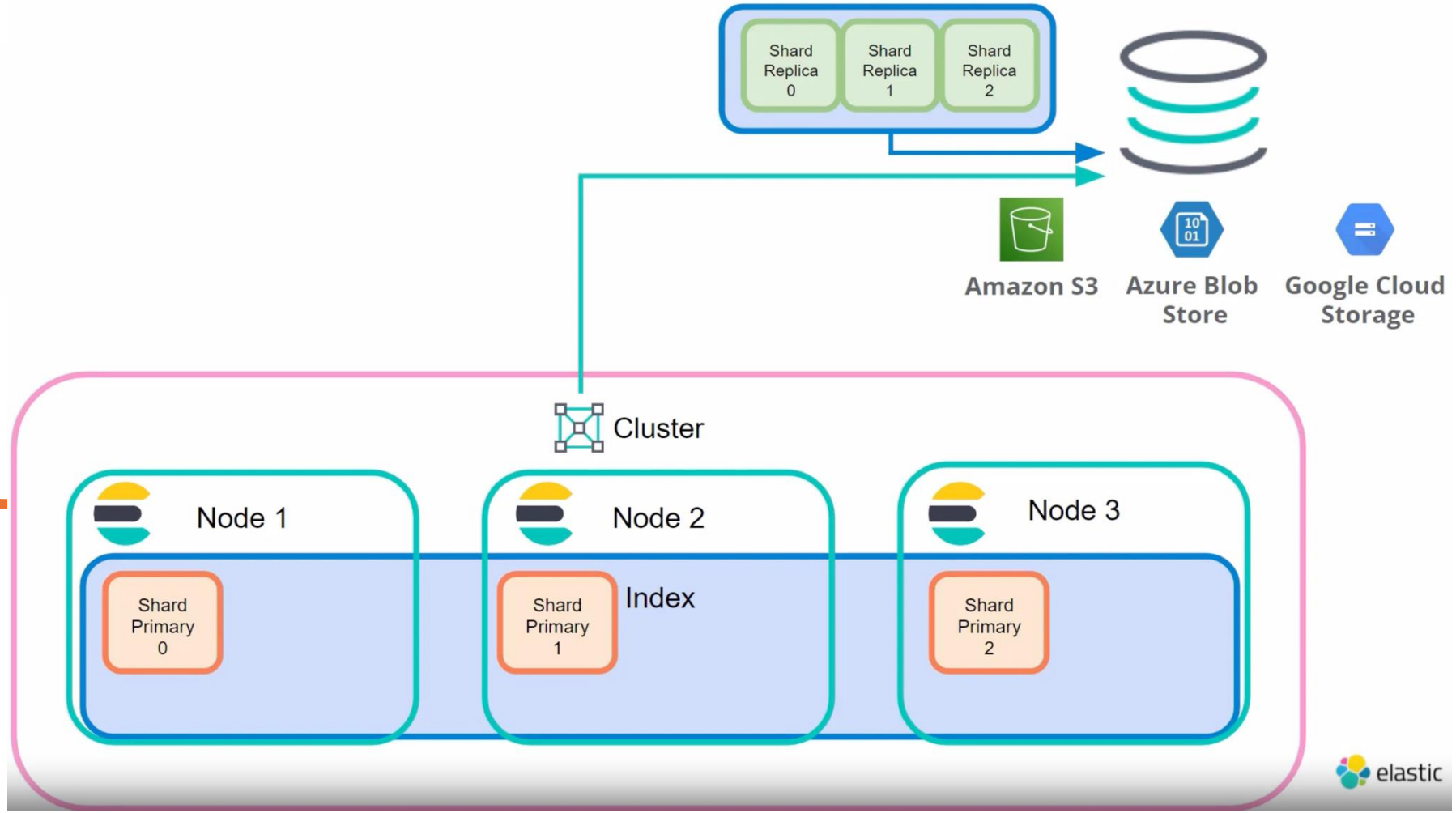


Data replication

- Trade offs:
  - Write scalability + Read reliability & availability
  - More replica shards => more resources: storage, CPU, & memory



Replica shards on the cloud



REST API

Elasticsearch: document store

- Elasticsearch provides REST APIs to communicate with a cluster over HTTP / HTTPS
  - enables to write your applications in any language (of course we'll be using Python)
  - use a language client to interact with Elasticsearch using language-native features