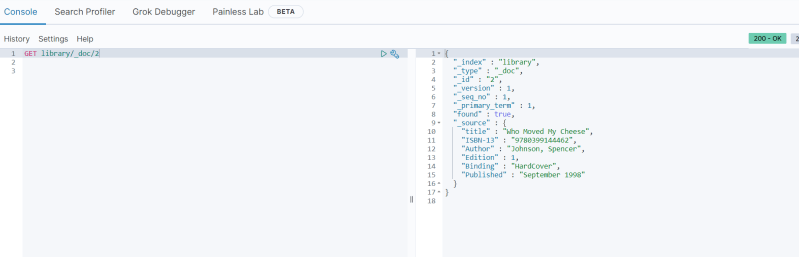
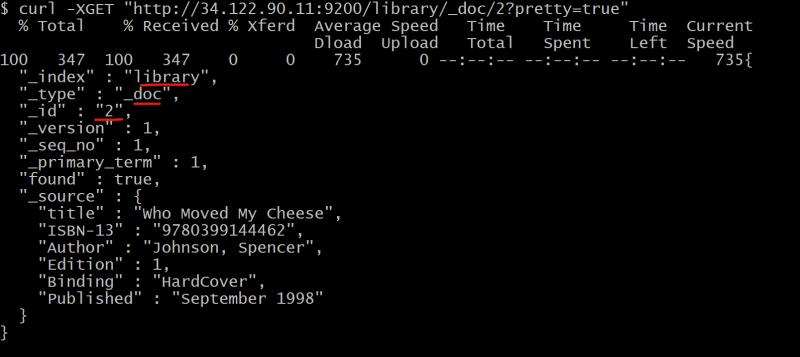
Documents

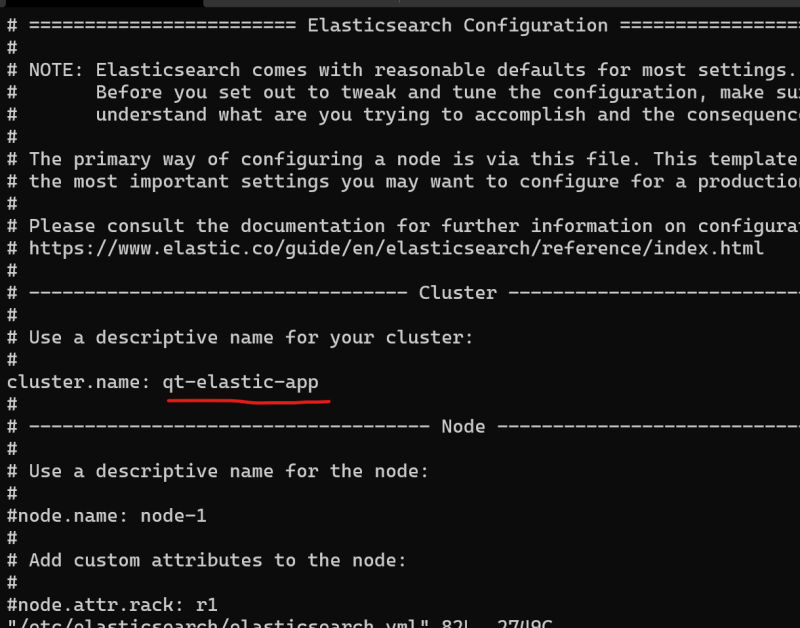
* JSON documents are first class citizens of Elastic Search.
* A document consists of multiple fields
* Each field in JSON document is of particular type
* In addition to the fields that are sent by user in the document , Elastic search adds the following metafields
  + \_id: This is unique identifier of the document with in the type
  + \_type: This field consists of type of the document
  + \_index: This field consists of index name of the document 

**Nodes**

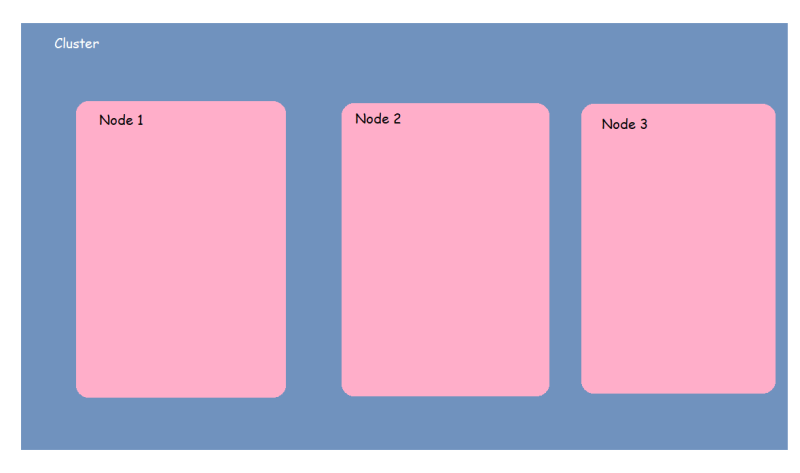
* Elastic search node is a single server of Elastic Search, which is part of large cluster of nodes.
* Nodes participate in indexing, searching and performing other operations supported by Elastic Search

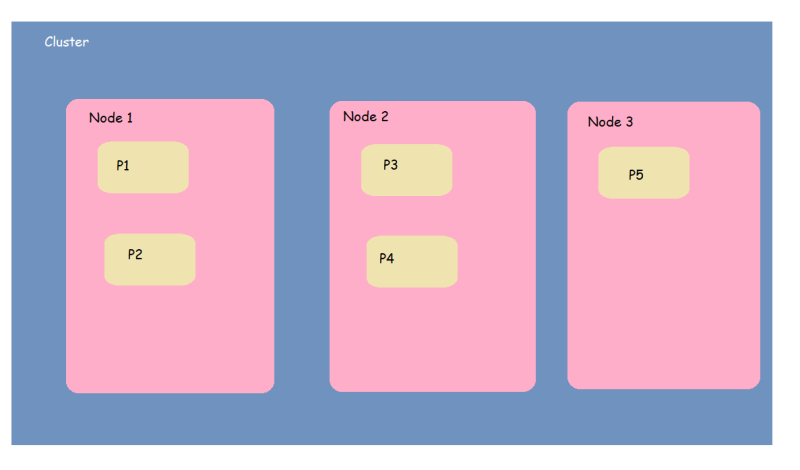
**Cluster**

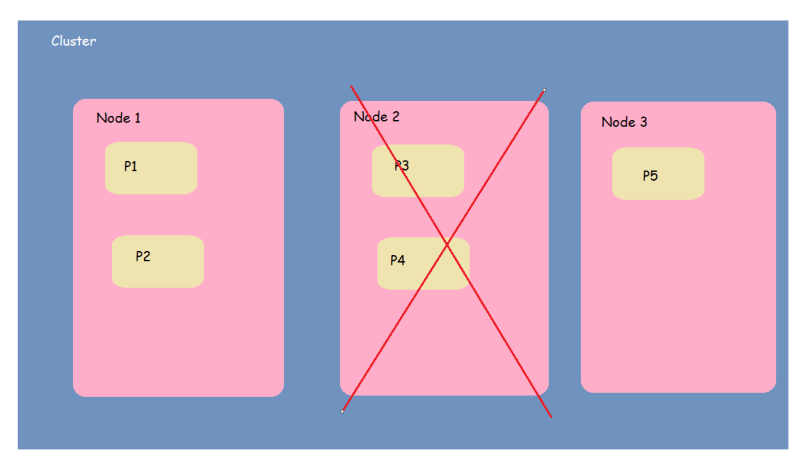
* A cluster hosts one or more indexes and is responsible for providing operations such as searching, indexing and aggregrations.
* A cluster can be formed by one or more nodes.

Every Elastic search node is always part of the cluster 

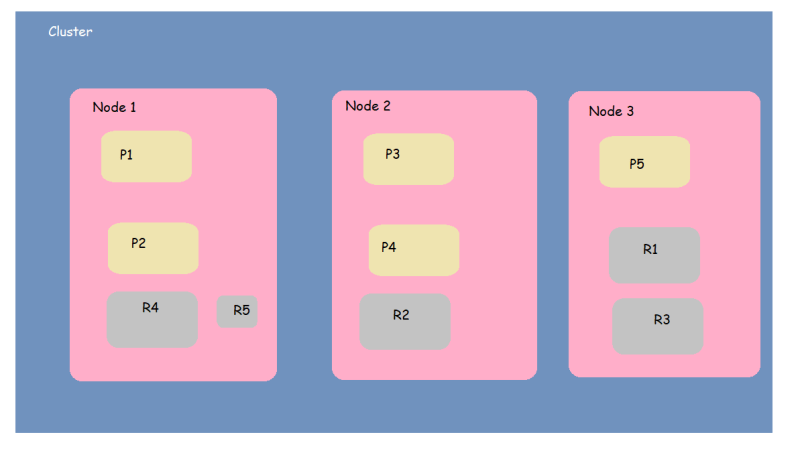
**Shards and replicas**

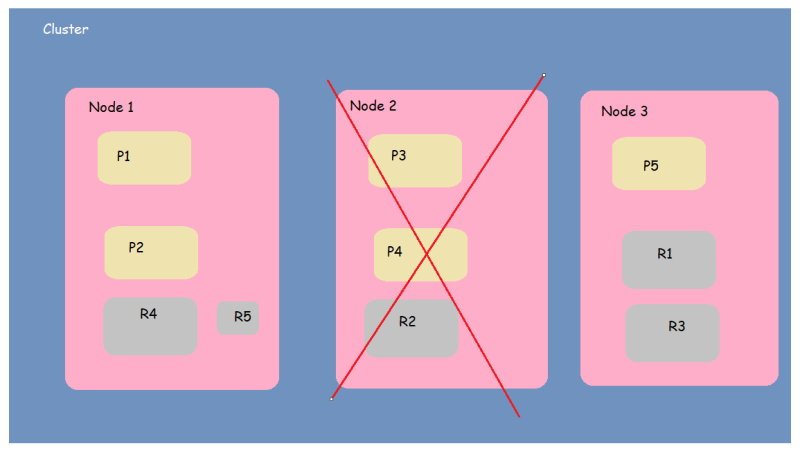
Let’s assume we have elastic search cluster with 3 nodes 

Shards help in distributing a index over the cluster (dividing the documents of single index over multiple nodes). This process of dividing the data among shards s called as sharding 

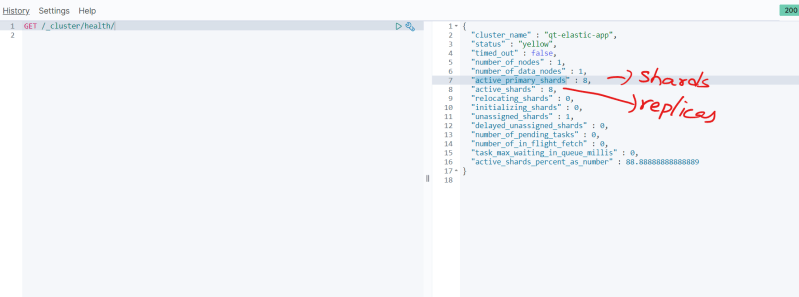
Now let’s imagine the Node 2 is failed 

Distribute systems such as elastic search are expected to run inspire of hardware failures, this is addressed by replica shards or replicas.

Each shard in an index can be configured to have zero or more replicas. 

Now let’s see what happens when one node i.e. node2 fails 

We still have the data available add more nodes will make elastic search more available and fault tolerant

By default, every index is configured to have five shards 

**Datatypes in Elastic Search**

* Core data types:
  + String data types
    - *text*: This datatype is used of supporting full-text search for fields that contain a description or lengthy text values
    - *keyword*: This type enables analytics of string fields. Fields of this type support sorting, filtering and aggregation
  + Numeric datatypes:
    - *byte, short, integer and long*
    - *float, double*
    - half\_float
    - scaled\_float
  + Date datatypes
    - *date*
  + Boolean datatype:
    - *boolean*
  + Binary datatype
    - *binary*
  + Range data types
    - integer\_range, float\_range, long\_range, double\_range and date\_range
* Complex data types
  + Array data type
  + Object data type
  + Nested Data type
* Other data types
  + Geo-point datatype
  + Geo-shape datatype
  + IP datatype

**Mapping**

* We have inserted two records into the elastic search the following tasks were performed by Elastic search
  + Creating an index with name library
  + Defining the mapping of the type of documents stored in the index’s default type \_doc
* Operations performed

PUT /library/\_doc/1

{

"title": "Mind Hacking, Unfck Yourself, Rich Dad Poor Dad, Smarter Faster Better 4 Books Collection Set",

"ISBN-10": "1612680178",

"Authors": [

"Sir John Hargrave", "Gary John Bishop", "Charles Duhigg", "Robert T. Kiyosaki"

],

"Edition": 2,

"Binding": "Paperback",

"List Price": "0.17$",

"Published": "January 2020"

}

PUT /library/\_doc/2

{

"title": "Who Moved My Cheese",

"ISBN-13": "9780399144462",

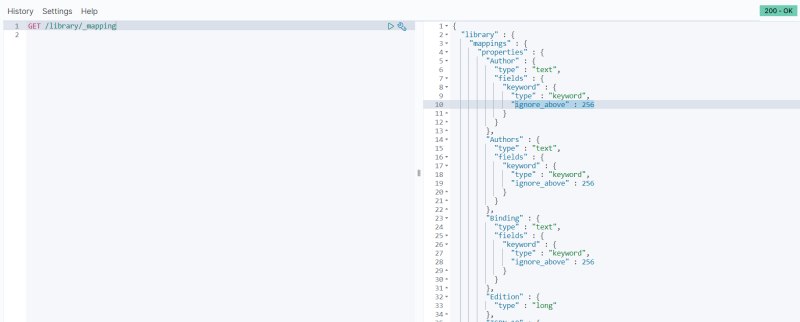
"Author": "Johnson, Spencer",

"Edition": 1,

"Binding": "HardCover",

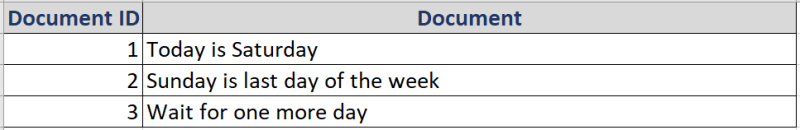
"Published": "September 1998"

}



* Elastic search automatically adds mapping to fields in document so that elastic search operations such as analyzing and aggregations can be done on this.

**Inverted Index**

* This is core data structure of Elastic Search and supports full text search.
* Lets assume we have 3 simple documents as shown below 
* Elasticsearch builds a datastructure from these 3 documents that have been index and this data structure is called as *inverted index*
* Notice the following
  + Documents were broken down into terms after removing punctuation and placing them in lowercase
  + Terms are sorted alphabetically
  + The Frequency column captures how many times the term appears in the entire document set
  + The third column captures the documents in which term was found

