

Every document has unique identifier .automatically or provided.

Documents are assigned to indexes.ex (customer data, sales data)

Another word on types

Phew, things sure are changing fast in regards to types! Not only are they being removed (which is great), but to prepare for version 7.x of Elasticsearch, you should use a slightly different syntax than what is used within most of this course as of today. Whenever you specify a type, you should use \_doc as the type name, because this will be the syntax that will be used in version 7.x. So by doing this, you will be future proofing your mappings and queries, ensuring that you won't face major headaches when upgrading Elasticsearch in the future.

At this point, you haven't seen any queries yet, but basically whenever you see me type default as the type, you should write \_doc instead. So when adding a document, for instance, you would see me type:

PUT /users/default/1

{

"name": "Bo Andersen"

}

Then you should type:

PUT /users/\_doc/1

{

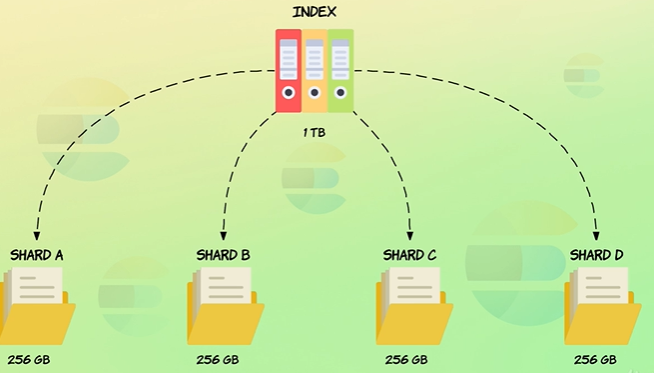
"name": "Bo Andersen"

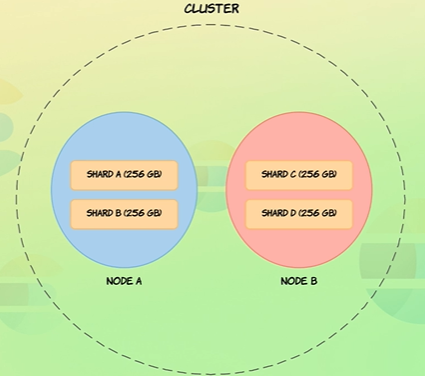
}

This will gradually be updated and reflected within the course, but please understand that re-recording and editing 10+ hours of content takes a very long time, so there will be a period in which you will see both types being specified.

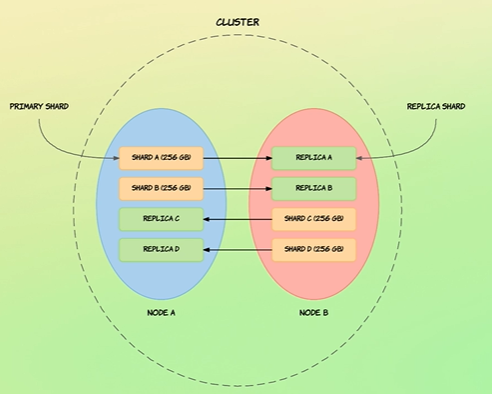
Note that you need to be running at least Elasticsearch 6.2.0 to be able to specify \_doc as the document type.

Thank you for your understanding, and happy searching!



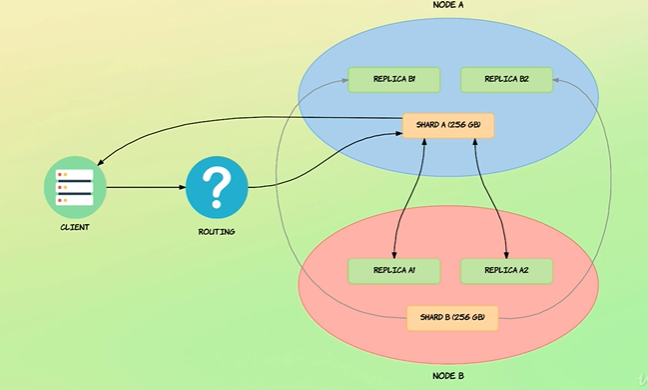


Sharding is a way of dividing index data volume in to multiple parts this enables to distribute data across multiple nodes. Ideally each node will be installed in different vms.

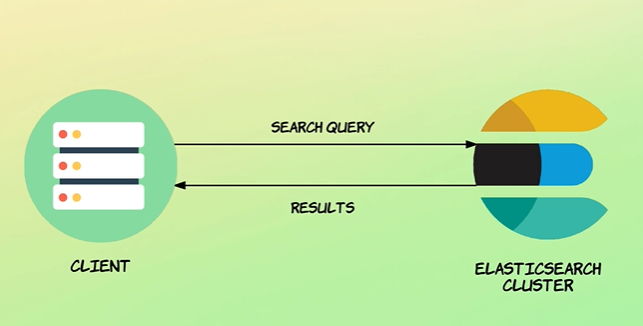


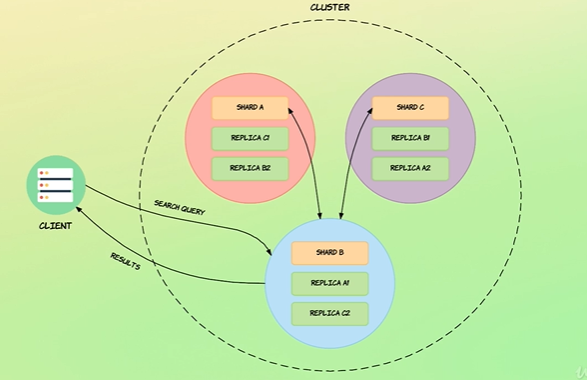
Primary shard gets replicated to replica shard. Write operation happens on primary shard.

Replicas will never be stored on same machine as its own shard. Replica is a copy of shard.

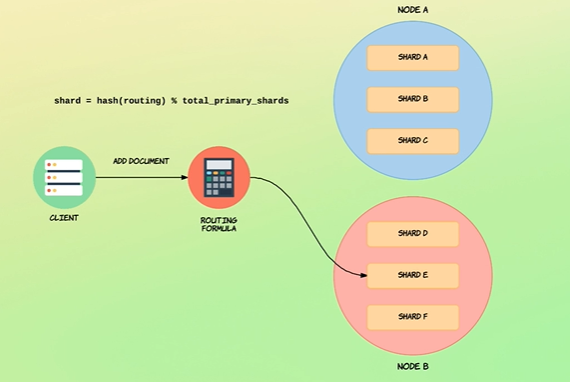


Any update, deletion or addition of data is done always on shard first then replicated in all the replicas.





Co-coordinating receives the request and sends it to other shards or replicas in other nodes and after receiving the result combines and sorts them returns to the client.

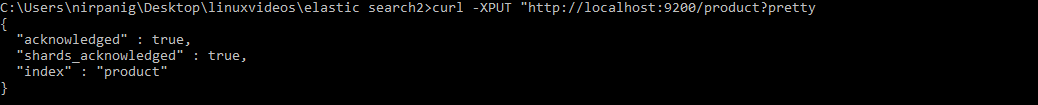


Number of shards cannot be changed once index has been created.

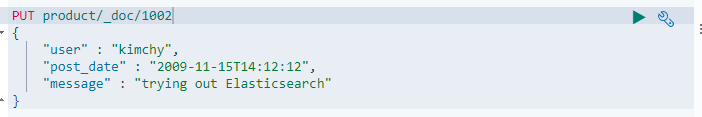
Elasticsearch.yml for configuring

**Creating index**

curl -XPUT "http://localhost:9200/product?pretty



**Creating a document**



curl -XPUT "http://localhost:9200/product/\_doc/1002" -H 'Content-Type: application/json' -d'{ "user" : "kimchy", "post\_date" : "2009-11-15T14:12:12", "message" : "trying out Elasticsearch"}'

**Searching Documents**

GET product/\_search 🡪

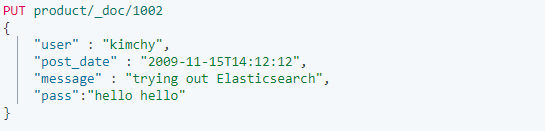
curl -XGET "http://localhost:9200/product/\_search"

GET product/\_doc/1 🡪 getting document based on id in kabana

curl -XGET "<http://localhost:9200/product/_doc/1>?pretty"

**Updating a document**

This way we can add a document or we can replace it with new document having extra field.



curl -XPUT "http://localhost:9200/product/\_doc/1002" -H 'Content-Type: application/json' -d'{ "user" : "kimchy", "post\_date" : "2009-11-15T14:12:12", "message" : "trying out Elasticsearch", "pass":"hello hello"}'



Each time the document is replaced its version number will increase.

**Updating specific fields in a document of elastic search**

In this approach we can update existing field’s value and also we can new fields to the document.

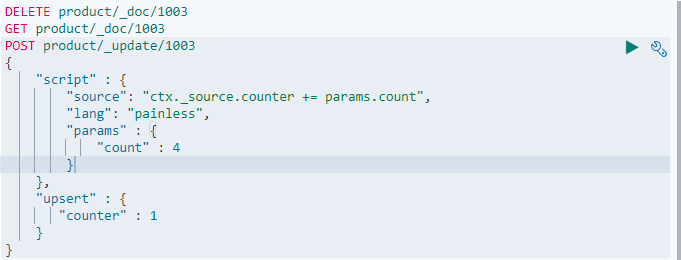


curl -XPOST "http://localhost:9200/product/\_update/1002?pretty" -H 'Content-Type: application/json' -d'{ "doc": { "user": "Jane Doee","price": 180 }}'

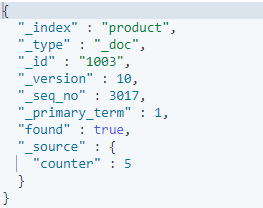
curl -XPOST "http://localhost:9200/product/\_update/1002?pretty" -H 'Content-Type: application/json' -d'{ "script" : "ctx.\_source.price += 5"}'

### Upsert

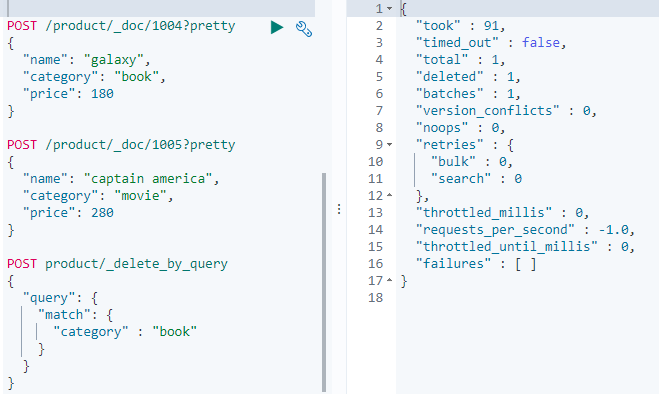
If the document does not already exist, the contents of the upsert element will be inserted as a new document. If the document does exist, then the script will be executed instead:



After running first time and second time the result given below.

**Delete By Query API**



**Deleting Index**



**Bulk API**

Using bulk api we can do multiple operation at the same time ie creating a document, updating multiple document or deleting documents.

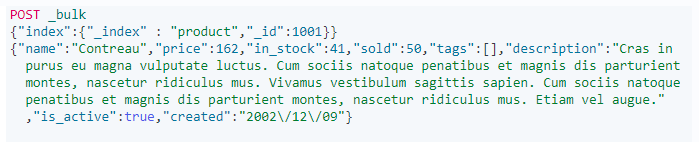


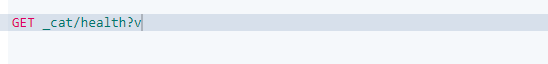
**Uploading bulk document**

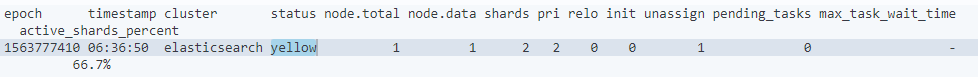


curl -s -H "Content-Type: application/x-ndjson" -XPOST localhost:9200/product/\_bulk?pretty --data-binary "@test-data.json"

Bulk posting example. Where \_index is the index under which we want to create a document.







Get information about cluster

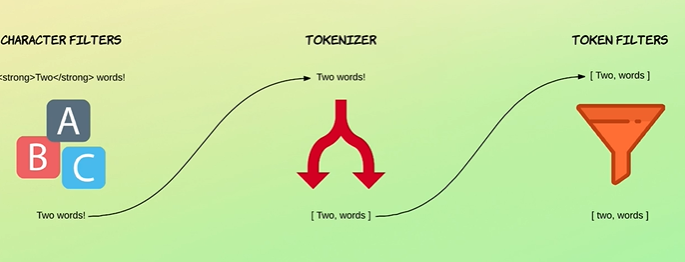
How to add mapping for document and add additional mapping after creating the mapping

Data types in elastic search

Need to write details about meta fields

In elastic search index mapping if dynamic is disabled and we created a document after that we are adding the mapping we will not able to filter it with that field.

We can reindex documents using update\_by\_query? conflicts=proceed



Character filter removes html mark up

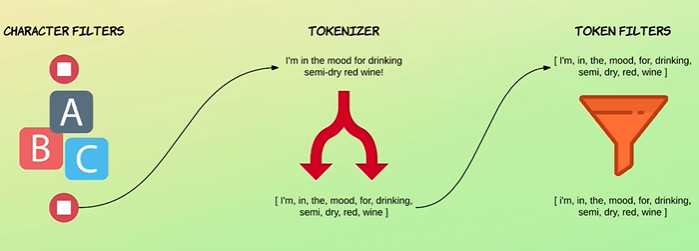
Tokenizer convers input multiple tokens based on white spaces removes commas semicolons hyphens.

Simplest Token filter convers the tokens in to lower case.

Stub removes common words the, an, a etc

Synonym token filter (nice, good)

**Standard Analyzer**

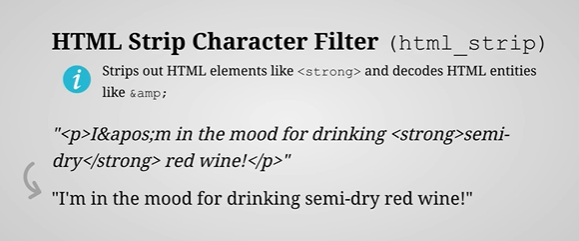


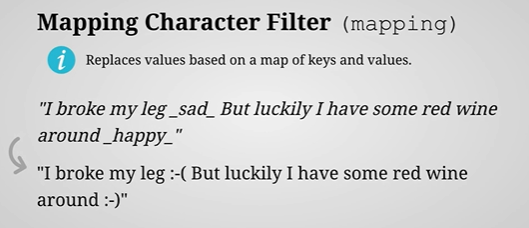
**Inverted Index**

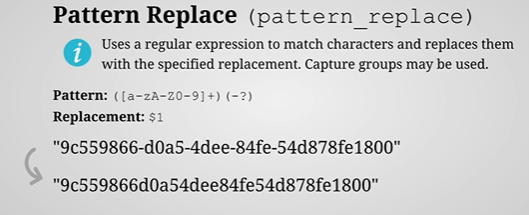
Inverted index is used for full text search. Analyzer is applied to full text fields and the results of this analysis is stored in an inverted index. An inverted is the collection of all the terms in all the documents



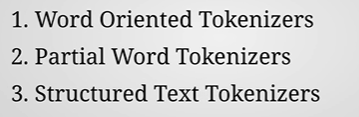
**Character Filter**



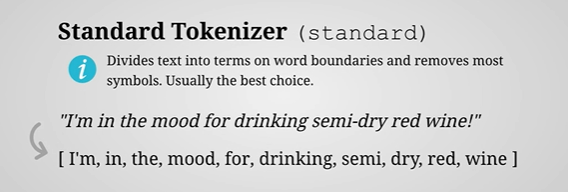


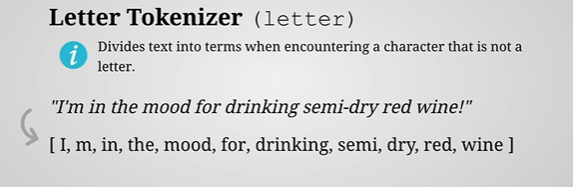


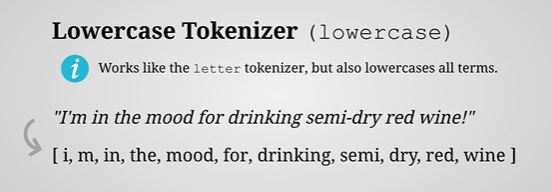
**Tokenizer**

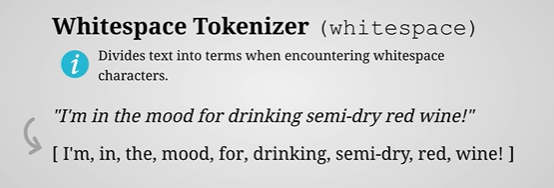








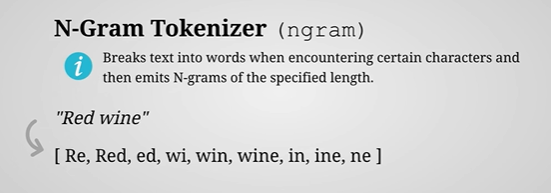


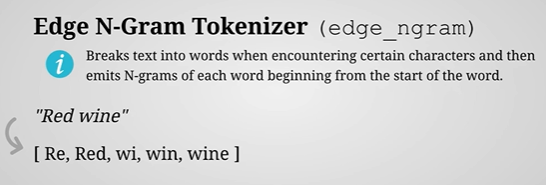




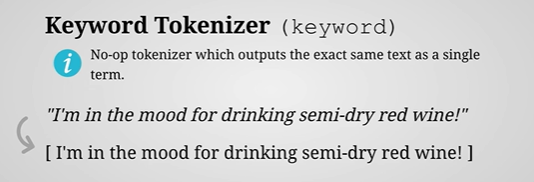
**Partial Word Tokenizer**

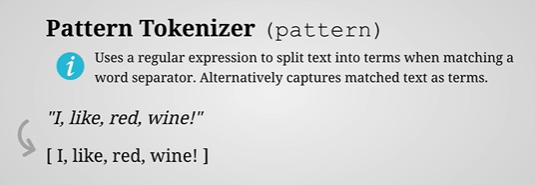


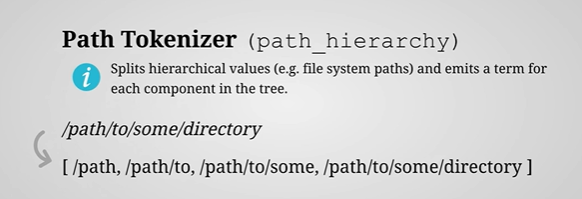






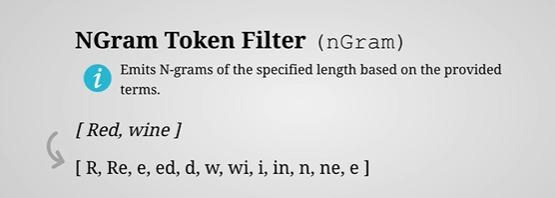


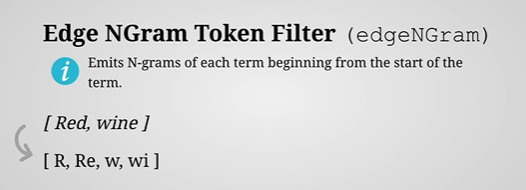




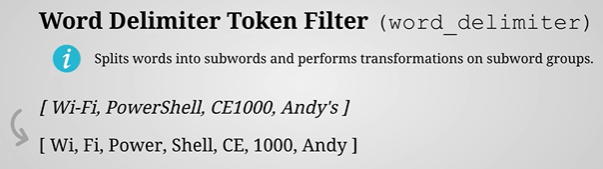
**Token Filters**



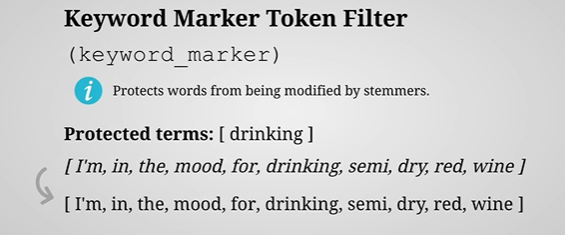








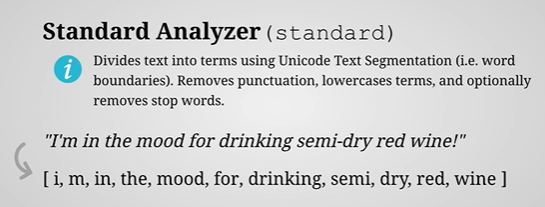


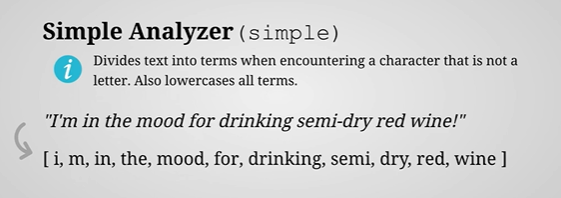


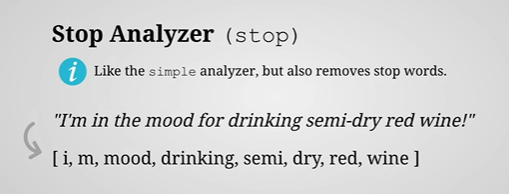


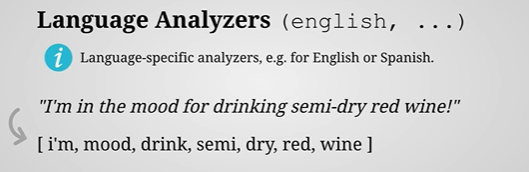


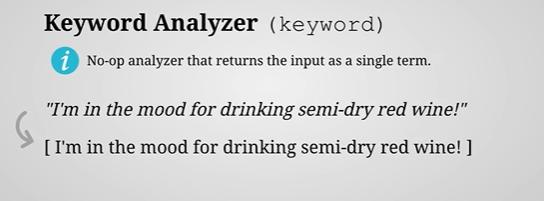
**Analyzers**



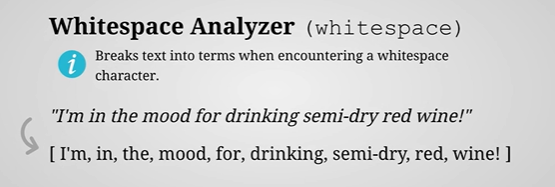




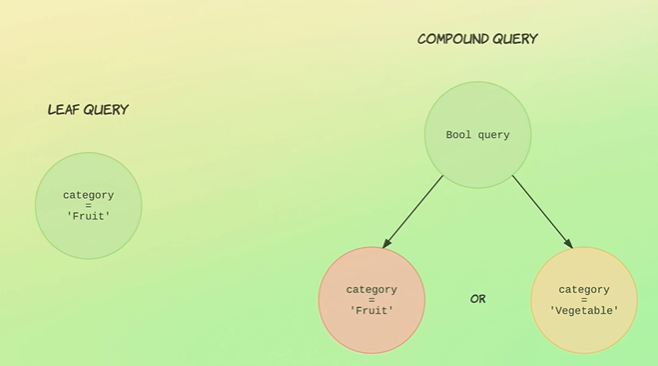








Analyzers can be applied at document level or field level. Field level overrides document level.



Query context calculates the relevance to how well the document match based on some algorithm. Filter context used whether the document contains the search field and that’s included in the result.

**Term level query and full text query**

Do not use term level query for full text search.

Term level queries are good for matching Boolean, dates, enums, and number .

