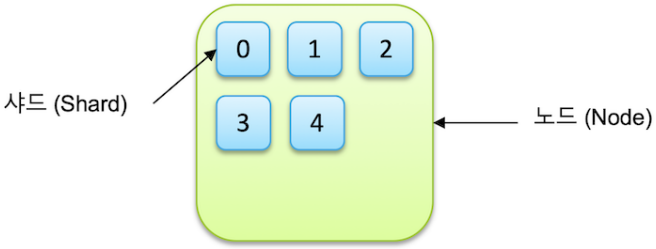
Elsaticsearch Cookbook

생성일	@2022년 12월 17일 오후 5:19
태그	

3장 기본 작업

- 색인 관리 방법 및 색인의 생성 / 조회 / 수정 / 삭제의 기능을 알아보자.
- 색인
  - SQL 데이터 베이스와 비슷한 구조를 가진 테이블이자 레코드 이다.
  - 단일 데이터의 단위를 나타내는 도큐먼트를 모아 놓은 집합
- 샤드
  - 분산 저장 단위로 각 노드에 샤드 단위로 데이터가 분산 저장



색인 생성

- HTTP method : `PUT`
- URL : `http://<serverUrl>/<indexName>`
- 예제
  - Request : `PUT http://localhost:9200/myindex`

```
{
    "settings" : {
        "index": {
            "number_of_shards": 2,
            "number_of_replicas" : 1
        }
    }
}
```

field	설명
number_of_shards	색인 구성 시 샤드 구성 수
number_of_replicas	고가용성을 위한 클러스터 수

- 응답

```
{
    "acknowledged": true,
    "shards_acknowledged": true,
    "index": "myindex"
}
```

- 에러

```
{
    "error": {
        "root_cause": [
            {
                "type": "resource_already_exists_exception",
                "reason": "index [myindex/ZzGqlxXgRZ6eauLehJ69Hg] already exists",
                "index_uuid": "ZzGqlxXgRZ6eauLehJ69Hg",
                "index": "myindex"
            }
        ],
        "type": "resource_already_exists_exception",
        "reason": "index [myindex/ZzGqlxXgRZ6eauLehJ69Hg] already exists",
        "index_uuid": "ZzGqlxXgRZ6eauLehJ69Hg",
        "index": "myindex"
    },
    "status": 400
}
```

- 내부 동작
  - 요청 색인이 첫 번째 노드에 생성된 후 클러스터에 속한 모든 색인에 상태 전파
  - 기본(empty) 매핑 생성
  - 모든 샤드의 초기화
- 제약사항
  - 아스키 문자 [a-z]
  - 숫자 [0-9]
  - 마침표 (.,) 하이픈 (-, &, _)
- 매핑 정의를 포함한 요청 `PUT` `http://localhost:9200/myindex`

```
{
    "settings": {
        "index": {
            "number_of_shards": 2,
            "number_of_replicas": 1
        }
    },
    "mappings": {
        "properties": {
            "id" : {
                "type": "keyword",
                "store": true
```

```
            },
            "date": {
                "type": "date",
                "store": false
            },
            "customer_id": {
                "type": "keyword",
                "store": true
            },
            "sent": {
                "type": "boolean"
            },
            "name": {
                "type": "text"
            },
            "quantity": {
                "type": "integer"
            },
            "vat": {
                "type": "double",
                "index": true
            }
        }
    }
}
```

색인 삭제

- HTTP method : `DELETE`
- URL : `http://<serverUrl>/<indexName>`
- 예제
  - Request : `PUT http://localhost:9200/myindex`
  - 응답

```
{
    "acknowledged": true
}
```

  - 에러

```
{
    "error": {
        "root_cause": [
            {
                "type": "index_not_found_exception",
                "reason": "no such index [myindex]",
                "resource.type": "index_or_alias",
                "resource.id": "myindex",
                "index_uuid": "_na_",
                "index": "myindex"
            }
        ],
        "type": "index_not_found_exception",
        "reason": "no such index [myindex]",
        "resource.type": "index_or_alias",
```

```
        "resource.id": "myindex",
        "index_uuid": "_na_",
        "index": "myindex"
    },
    "status": 404
}
```


- 내부 동작
  - 모든 색인 데이터가 디스크에서 삭제
    - 클러스터 갱신
    - 스토리지에서 샤드 삭제
- 전체 인덱스 삭제 : `DELETE http://localhost:9200/_all`
  - 전체 삭제 비활성화
    - `elasticsearch.yml`
    - `action.destructive_requires_name : ture`

색인 여닫기

- 데이터 보존과 자원 절약을 위한 삭제하기의 대안
- 온라인 오프라인 모드로 전환
- HTTP method : `POST`
- HTTP URL
  - 닫기 : `http://<serverUrl>/<indexName>/_close`
  - 열기 : `http://<serverUrl>/<indexName>/_open`
- 닫기 예제
  - Request : `POST http://localhost:9200/_close`
  - 닫은 후 동작 시 에러 반환


```
{
    "error": {
        "root_cause": [
            {
                "type": "index_closed_exception",
                "reason": "closed",
                "index_uuid": "PBGZKO-dT8igX6E9hh1Sow",
                "index": "myindex"
            }
        ],
        "type": "index_closed_exception",
        "reason": "closed",
        "index_uuid": "PBGZKO-dT8igX6E9hh1Sow",
        "index": "myindex"
    },
    "status": 400
}
```

- 열기 예제
  - Request : `POST http://localhost:9200/_open`
- 내부 동작
  - index 닫기 수행 시에 해당 인덱스에 대한 READ / WRITE 등 모든 수행 요청이 되지 않음
    - 색인 샤드가 오프라인으로 전환되어 파일 디스크립터, 메모리, 스레드를 사용하지 않음
  - 여닫기 수행 시에 마스터는 인덱스의 상태 변경을 위해 인덱스 샤드를 재시작한다.

색인에 매핑 집어넣기

- DB의 테이블 생성과 비슷한 작업
- HTTP method : `PUT`, `POST`
- URL : `http://<serverUrl>/<indexName>/_mapping`
- 예제
  - Request : `PUT` `http://localhost:9200/myindex/_mappings`

```
{
    "properties": {
        "id" : {
            "type": "keyword",
            "store": true
        },
        "date": {
            "type": "date",
            "store": false
        },
        "customer_id": {
            "type": "keyword",
            "store": true
        },
        "sent": {
            "type": "boolean"
        },
        "name": {
            "type": "text"
        },
        "quantity": {
            "type": "integer"
        },
        "vat": {
            "type": "double",
            "index": true
        }
    }
}
```

- 내부 동작
  - 색인이 존재하는지 확인 후 매핑 유형을 생성

    - 이미 존재하는 유형일 경우 병합
    - 필드의 유형이 기존과 다르다면 해당 유형은 갱신되지 않고, 필드 속성 확장 예외가 발생

```
{
    "error": {
        "root_cause": [
            {
                "type": "illegal_argument_exception",
                "reason": "mapper [price] cannot be changed from type [integer] to [keyword]"
            }
        ],
        "type": "illegal_argument_exception",
        "reason": "mapper [price] cannot be changed from type [integer] to [keyword]"
    },
    "status": 400
}
```

        - 병합되길 원한다면, `ignore_conflicts = true` 로 지정한다.
- 매핑 삭제
    - 매핑의 삭제는 되지 않음
    - 매핑의 삭제를 원한다면 다음의 동작을 따라야 함
        - 신규 또는 변경된 매핑으로 신규 색인 생성
        - 모든 레코드를 재색인
        - 매핑이 잘못된 이전 색인 삭제
    - 5.x 이후 버전은 `reindex` 명령을 수행

매핑 가져오기

- HTTP method : `GET`
- URL
    - `http://<serverUrl>/_mapping`
    - `http://<serverUrl>/<indexName>/_mapping`
- 예제
    - Request : `GET http://localhost:9200/myindex/_mappings`
    - 모든 mapping 가져오기

```
GET /*/_mapping

GET /_all/_mapping

GET /_mapping
```

    - 다중 조회 : `GET /my-index-000001,my-index-000002/_mapping`

- 내부 동작
  - 색인과 유형의 존재 여부를 체크한 후 저장된 매핑을 반환한다.

색인 재색인

- 매핑 변경을 위한 재색인 수행이 필요한 경우가 있음
  - 매핑에 새로운 부속 필드를 추가하면, 새로운 부속 필드 검색을 위해 레코드 재처리를 수행
  - 사용하지 않는 매핑 제거
  - 신규 매핑을 필요로 하는 레코드 구조로 변경
- HTTP method : `POST`
- URL : `http://<serverUrl>/_reindex`
- 예제
  - Request : `http://localhost:9200/_reindex?pretty=true`

```
{
    "source": {
        "index": "myindex"
    },
    "dest": {
        "index": "myindex2"
    }
}
```

  - 응답

```
{
    "took": 29,
    "timed_out": false,
    "total": 0,
    "updated": 0,
    "created": 0,
    "deleted": 0,
    "batches": 0,
    "version_conflicts": 0,
    "noops": 0,
    "retries": {
        "bulk": 0,
        "search": 0
    },
    "throttled_millis": 0,
    "requests_per_second": -1.0,
    "throttled_until_millis": 0,
    "failures": []
}
```

  - 내부 동작
    - 엘라스틱서치 작업의 초기화
    - 대상 색인 생성 및 원본 매핑 복사

        - 재색인될 문서를 수집하는 쿼리 실행
        - 모든 문서가 재색인될 때까지 대규모로 문서 재색인
    - 주의 사항
        - 재색인은 copy의 동작이 아니다.
    - 추가 동작
        - remote cluster에 대한 재색인
            - source에 `remote` 정보 추가
        - 일부 field만 재색인
        - 일부 데이터만 재색인

색인 새로 고침

- 색인을 강제로 새로 고침하여 검색 상태를 제어
- HTTP method : `POST`
- URL : `http://<serverUrl>/<indexName>/_refresh`
- 예제
    - Request: `http://localhost:9200/myindex/_refresh`
    - 응답

```
{
    "_shards": {
        "total": 4,
        "successful": 2,
        "failed": 0
    }
}
```

    - 동작
        - 새로 고침 간격을 통해 자동으로 새로고침 되지만 해당 간격이전에 새로 고침하게 된다.
        - 일반적으로는 대량의 데이터를 색인한 뒤에 저장된 레코드의 즉시 검색을 보장하기 위해 사용한다.,

색인 청소

- elasticsearch는 디스크 I/O 오버헤드를 줄이고자 갱신 발생 전까지 일부 데이터를 메모리와 트랜잭션 로그에 저장한다.
- 메모리 해제를 위해 트랜잭션 로그를 비워 데이터를 디스크에 기록되도록 보장한다.
- HTTP method : `POST`
- URL : `http://<serverUrl>/<indexName>/_flush[?refresh=True]`

- 요청
  - request : `http://localhost:9200/myindex/_flush`
  - 응답

```
{
    "_shards": {
        "total": 4,
        "successful": 2,
        "failed": 0
    }
}
```

색인 강제 병합

- elasticsearch는 디스크의 세그먼트에 데이터를 저장한다.
- 세그먼트 수의 증가에 따라 검색 속도가 저하된다.
- HTTP method : `POST`
- URL : `http://<serverUrl>/<indexName(s)>/_flush[?refresh=True]`
- 예제
  - Request : `http://localhost:9200/myindex/_forcemerge`
  - 응답

```
{
    "_shards": {
        "total": 4,
        "successful": 2,
        "failed": 0
    }
}
```

- 동작
  - elasticsearch는 delete된 문서에 대해 논리적 제거(delete 마킹)를 수행한다
  - 여유 공간 확보를 위해 강제 병합을 시도한다. 이 과정에서 세그먼트 수가 커질 수 있다.
  - 루씬의 강제 병합 작업은 사용하지 않는 세그먼트 제거 및 삭제된 문서를 제거하여 최소의 세그먼트를 갖도록 색인을 재생성 하는 등 I/O 오버헤드를 유발하며 세그먼트 축소를 시도한다.

색인 축소

- 색인의 샤드를 축소하기 위한 방법
- 예시
  - 초기 설계 당시 샤드 숫자를 잘못 계산한 경우
  - 메모리와 자원 샤용량을 줄이고 싶을 때

    - 검색 속도를 높이고자 할 때
- 샤드 축소 하기전 체크사항
    - index는 반드시 `read-only`
    - 모든 기본 샤드는 동일 노드에 존재해야한다.
    - 인덱스는 반드시 green 상태여야 함
    - 샤드의 할당을 쉽게 하기위해서 replica를 모두 제거하는 걸 추천
- HTTP method : `POST`
- URL : `http://<serverUrl>/<sourceIndexName>/_shrink/<targetIndex>`
- 예제
    - Request : `http://localhost:9200/myindex/_shrink/reduced_index`

```
{
    "settings" : {
        "index": {
            "number_of_shards": 1,
            "number_of_replicas" : 1
        }
    }
}
```

- 동작 방식
    - 원본 index와 동일한 정의를 가진 새로운 index를 생성한다.하지만 기존 샤드 갯수와 다르게 적은 수의 기본 샤드만 생성된다.
    - 원본 index의 세그먼트를 새롭게 생성된 index에 하드링크 한다.
        - 만약, 파일 시스템이 하드링크를 지원하지 않는다면, 새로운 인덱스에 전체 복사 하며, 오랜 시간을 소요한다.
        - 다중 데이터 경로를 사용한다면, 하드링크가 디스크에서 동작하지 않으므로 서로 다른 데이터 경로에 있는 세그먼트의 전체 복사본이 필요하다.

색인 존재 확인

- 색인의 존재 유무를 확인하는 방법으로 HTTP URL 존재를 확인할 때 HEAD를 이용해 확인하는 방법과 동일한 방법.
- HTTP method : `HEAD`
- URL : `http://<serverUrl>/<indexName>`
- 예제
    - Request : `http://localhost:9200/myindex`

색인 설정 관리

- 가져오기
  - HTTP method : `GET`
  - URL : `http://<serverUrl>/_settings?pretty=true`
  - 예제
    - Request : `http://localhost:9200/myindex/_settings?pretty=true`

```
{
    "myindex": {
        "settings": {
            "index": {
                "routing": {
                    "allocation": {
                        "include": {
                            "_tier_preference": "data_content"
                        }
                    }
                },
                "number_of_shards": "2",
                "provided_name": "myindex",
                "creation_date": "1671268247326",
                "number_of_replicas": "1",
                "uuid": "PBGZKO-dT8igX6E9hh1Sow",
                "version": {
                    "created": "7170499"
                }
            }
        }
    }
}
```

- 색인 변경
  - HTTP method : `PUT`
  - URL: `http://<serverUrl>/myindex/_settings`
  - 예제
    - Request: `http://localhost:9200/myindex/_settings`

```
{
    "index": {
        "number_of_replicas" : "2"
    }
}
```

```
{
    "acknowledged": true
}
```

색인 별칭

- 데이터가 다중 색인에 저장된 경우 간단하게 별칭을 이용해 색인을 관리 할 수 있다.
- 별칭 생성
  - HTTP method : `PUT`
  - URL : `http://<serverUrl>/_aliases/myalias1`
  - 예제
    - Request: `http://localhost:9200/myindex/_alias/myalias`
- 별칭 가져오기
  - HTTP method : `GET`
  - URL
    - `http://<serverUrl>/_aliases`
    - `http://<serverUrl>/<indexName>/_alias`
  - 예제
    - Request : `http://localhost:9200/myindex/_alias`

```
{
    "myindex": {
        "aliases": {
            "myalias": {}
        }
    }
}
```

    - Request : `http://localhost:9200/myindex/_alias`

색인 롤링

- 로그를 롤링하는 것과 같이 색인에도 롤링을 적용할 수 있다.
- HTTP method : `POST`
- URL : `http://<serverUrl>/<alias>/_rollover`
- 예제
  - Request : `http://localhost:9200/logs_writer/_rollover`
  - RequestBody

```
{
     "conditions": {
         "max_age": "7d",
         "max_docs": 10000
     },
     "settings": {
         "index.number_of_shards": 2
     }
}
```

```
{
    "acknowledged": false,
    "shards_acknowledged": false,
    "old_index": "mylogs-00001",
    "new_index": "mylogs-000002",
    "rolled_over": false,
    "dry_run": false,
    "conditions": {
        "[max_docs: 10000]": false,
        "[max_age: 7d]": false
    }
}
```

문서 색인

- 단일 또는 다중 문서를 색인에 저장하는 의미
- 관계형 데이터베이스에 레코드를 삽입하는 개념
- HTTP method : PUT, POST
- URL : `http://<serverUrl>/<indexName>/_doc/2qwlkdqlwkdqdkql`

Method	URL
POST	`http://<serverUrl>/<indexName>/_doc`
PUT/POST	`http://<serverUrl>/<indexName>>/_doc/<id>`
PUT/POST	`http://<serverUrl>/<indexName>>/_doc/<id>/_create`

- 예제
  - Request : `POST http://localhost:9200/myindex/_doc`
  - RequestBody

```
{
    "id" : "1234",
    "date": "2022-10-12T12:34:56",
    "customer_id": "customer1",
    "sent": true,
    "name": "customer",
    "in_stock_items": 0,
    "items": [
        {
            "quantity": 1,
            "vat": 1000,
            "price": 20000
        }
    ]
}
```

```
{
    "_index": "myindex",
    "_type": "_doc",
```

```
    "_id": "M1c2Q4UBiBbMu2pHjm3w",
    "_version": 1,
    "result": "created",
    "_shards": {
        "total": 2,
        "successful": 1,
        "failed": 0
    },
    "_seq_no": 0,
    "_primary_term": 5
}
```

        - _id: path에 ID를 지정하지 않아서 자동 생성된 ID
        - _version: 낙관적 동시성 제어에 따라 색인된 문서 버전
        - _result: 레코드 생성 여부
    - 파라미터
        - `POST /myindex/_doc?routing=1`
            - 색인에 사용되는 샤드에게 문서 색인
        - `POST /myindex/_doc?consistency=true`
            - 샤드의 구성 선택 (one/quorum/all)
        - `POST /myindex/_doc?replication=async`
            - 복제본 그룹의 복제 방식 설정
- 내부 동작
    - ID를 이용하여, 라우팅 또는 상위 메타데이터를 기반으로 올바른 샤드로 라우팅한다.
        - 없을 경우에는 새로운 ID를 생성하여 라우팅한다.
    - JSON 데이터를 확인하고 새로운 필드가 있다면 맵핑에 새로 추가된다.
    - 샤드에 문서를 색인한다. 이미 존재하는 ID가 있다면 갱신한다.
- 주의 사항
    - 성능을 위해 ID는 동일 길이로 하는 것이 좋다 (ID를 저장하는 데이터 트리의 균형을 위해)
    - REST 호출의 URL 인코딩 및 디코딩으로 인해 ASCII 사용하지 않으면 주의 해야한다.

문서 가져오기

- 새로고침 없이 실시간으로 문서를 가져온다.
- HTTP method : `GET`
- URL : `http://<serverUrl>/_doc/<id>`
- 예제
    - Request : `GET http://localhost:9200/_doc/M1c2Q4UBiBbMu2pHjm3w`
    - ReponseBody

```
{
    "_index": "myindex",
    "_type": "_doc",
    "_id": "M1c2Q4UBiBbMu2pHjm3w",
    "_version": 1,
    "_seq_no": 0,
    "_primary_term": 5,
    "found": true,
    "_source": {
        "id": "1234",
        "date": "2022-10-12T12:34:56",
        "customer_id": "customer1",
        "sent": true,
        "name": "customer",
        "in_stock_items": 0,
        "items": [
            {
                "quantity": 1,
                "vat": 1000,
                "price": 20000
            }
        ]
    }
}
```


- 내부동작
  - 모든 GET 호출은 실시간이다.
  - 엘라스틱서치는 빠른 조회를 위해 검색 문서를 포함한 샤드로만 리다이렉트한다.
  - 문서의 ID를 종종 메모리에 캐싱하기 때문에 호출이 매우 빠르다.
- 추가 사항
  - 소스만 가져오기
    - `GET http://localhost:9200/_doc/M1c2Q4UBiBbMu2pHjm3w/_source`
    - ResponseBody


```
{
    "id": "1234",
    "date": "2022-10-12T12:34:56",
    "customer_id": "customer1",
    "sent": true,
    "name": "customer",
    "in_stock_items": 0,
    "items": [
        {
            "quantity": 1,
            "vat": 1000,
            "price": 20000
        }
    ]
}
```


문서 삭제

- HTTP method : `DELETE`
- URL : `http://<serverUrl>/_doc/<id>`
- 예제
  - Request : `http://localhost:9200/myindex/_doc/M1c2Q4UBiBbMu2pHjm3w`
  - ResponseBody

```
{
    "_index": "myindex",
    "_type": "_doc",
    "_id": "M1c2Q4UBiBbMu2pHjm3w",
    "_version": 4,
    "result": "deleted",
    "_shards": {
        "total": 2,
        "successful": 1,
        "failed": 0
    },
    "_seq_no": 3,
    "_primary_term": 6
}
```

  - 내부 동작
    - 문서를 포함하고 있는 샤드만 접근하여 오버헤드가 없다.
    - 자식 문서인 경우에는 부모를 지정해야 한다.
    - 삭제하면 모든 문서는 영원히 지워진다.
    - id가 없을 경우 `delete_by_query` 를 사용해야한다.

문서 갱신

- HTTP method : `POST`
- URL : `http://<serverUrl>/<indexName>/_update/<id>`
- 예제
  - Request : `http://localhost:9200/myindex/_update/M1c2Q4UBiBbMu2pHjm3w`
  - RequestBody

```
{
    "script": {
        "source": "ctx._source.in_stock_items += params.count",
        "params": {
            "count": 4
        }
    }
}
```

```
{
    "_index": "myindex",
    "_type": "_doc",
    "_id": "M1c2Q4UBiBbMu2pHjm3w",
    "_version": 3,
    "result": "updated",
    "_shards": {
        "total": 2,
        "successful": 1,
        "failed": 0
    },
    "_seq_no": 2,
    "_primary_term": 6
}
```

- 변경 확인

```
{
    "_index": "myindex",
    "_type": "_doc",
    "_id": "M1c2Q4UBiBbMu2pHjm3w",
    "_version": 3,
    "_seq_no": 2,
    "_primary_term": 6,
    "found": true,
    "_source": {
        "id": "1234",
        "date": "2022-10-12T12:34:56",
        "customer_id": "customer1",
        "sent": true,
        "name": "customer",
        "in_stock_items": 8,
        "items": [
            {
                "quantity": 1,
                "vat": 1000,
                "price": 20000
            }
        ]
    }
}
```

- 내부동작
  - 문서를 가져와 기술된 대로 변경 후에 변경사항을 반영해 재색인한다.
  - 스크립트는 ctx Map 변수를 이용해 업데이트 할 수 있다.
    - _index
    - _type
    - _id
    - _version
    - _routing
    - _now (_timestamp)

        - op
    - 추가 동작
        - 필드 추가

```
POST test/_update/1
{
  "script" : "ctx._source.new_field = 'value_of_new_field'"
}
```

        - 필드 삭제

```
POST test/_update/1
{
  "script" : "ctx._source.remove('new_field')"
}
```

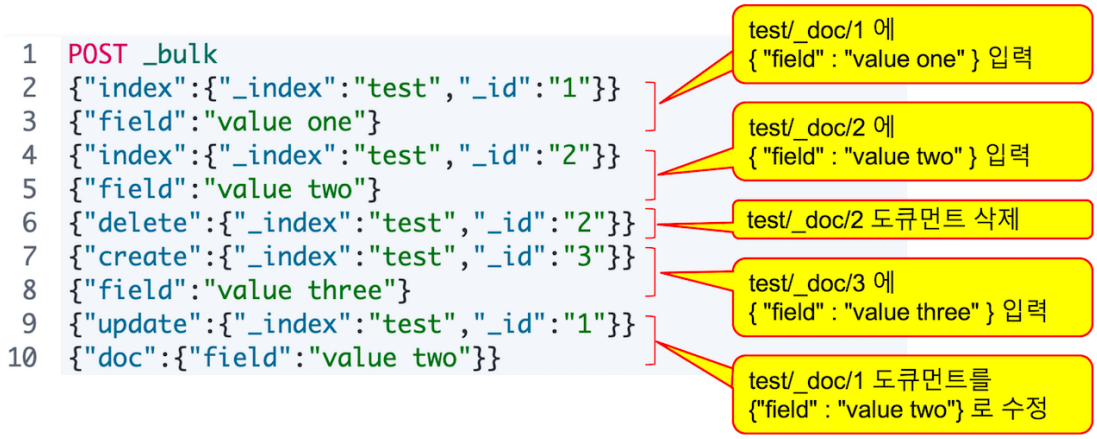
        -

단위 작업 속도 올리기(벌크 작업)

- 대량의 문서를 삽입, 삭제 또는 갱신할 때 벌크 요청하여 오버헤드를 줄인다.
- HTTP method : `POST`
- URL : `http://<serverUrl>/<indexName>/_bulk`
- 예제
    - Request : `http://localhost:9200/myindex/_bulk`
    - RequestBody

```
{ "index" : { "_index" : "test", "_id" : "1" } }
{ "field1" : "value1" }
{ "delete" : { "_index" : "test", "_id" : "2" } }
{ "create" : { "_index" : "test", "_id" : "3" } }
{ "field1" : "value3" }
{ "update" : {"_id" : "1", "_index" : "test"} }
{ "doc" : {"field2" : "value2"}}
```

    - 내부 동작



- 엘라스틱서치는 rollback이나 commit등의 트랜잭션을 지원하지 않아 bulk 작업 중 문제가 생기면 어디까지 실행되었는지 확인이 불가능 하므로, 전체 인덱스를 삭제 후 다시 시도하는 것이 안전하다.

GET 작업 속도 올리기(다중 GET)

- HTTP method : `GET`
- URL : `http://<serverUrl>/_mget`
- 예제
  - Request : `http://localhost:9200/_mget`
  - RequestBody

```
{
  "docs": [
    {
      "_index": "myindex",
      "_id": "M1c2Q4UBiBbMu2pHjm3w"
    },
    {
      "_index": "myindex",
      "_id": "2"
    }
  ]
}
```

```
{
    "docs": [
        {
            "_index": "myindex",
            "_type": "_doc",
            "_id": "682ZToUBF6fobCMdGMQt",
            "_version": 1,
            "_seq_no": 0,
            "_primary_term": 6,
            "found": true,
            "_source": {
                "id": "1234",
                "date": "2022-10-12T12:34:56",
```

```
                "customer_id": "customer1",
                "sent": true,
                "name": "customer",
                "in_stock_items": 0,
                "items": [
                    {
                        "quantity": 1,
                        "vat": 1000,
                        "price": 20000
                    }
                ]
            }
        },
        {
            "_index": "myindex",
            "_type": "_doc",
            "_id": "2",
            "found": false
        }
    ]
}
```