

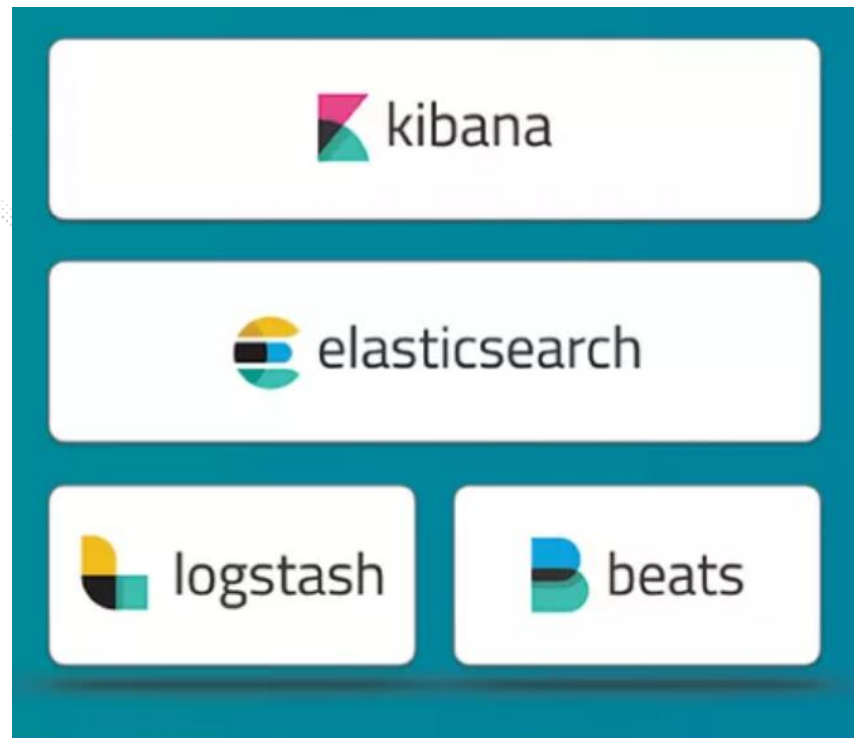
Elastic Stack

Kim Hye Kyung
topickim@naver.com

| Elastic Stack





Elastic Stack이란?

- ELK(ELK Stack)
 - Elasticsearch + Logstash + Kibana를 같이 묶어 서비스명으로 제공
 - 대용량 데이터 저장 및 빠른 검색
 - 데이터 분석
 - 데이터 시각화
 - 훌륭한 검색 엔진 및 시각화 tool 기능
- Elastic Stack history
 - 5.0.0 버전부터 Beats가 포함
 - **ElasticSearch + Logstash + Kibana + Beats**



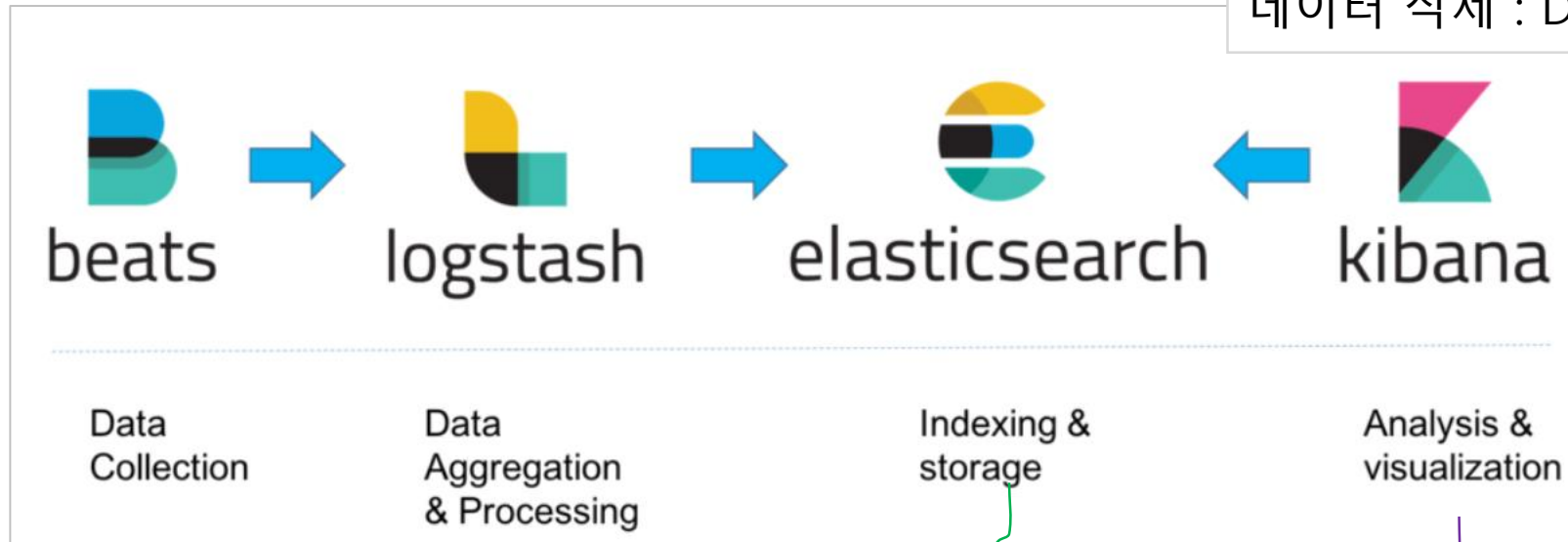
Elastic Stack

Elastic Stack 주요 기능

명 칭	설 명
ElasticSearch 	데이터 검색, 분석, 저장 많은 양의 데이터를 보관하고 분산형, 실시간으로 분석 엔진 데이터 저장소, 빅데이터 처리할 때 매우 유용(RDBMS 와 흡사한 기능) 확장성(json 문법만 OK)이 매우 좋은 자바로 구현된 오픈소스 검색엔진
Beats 	데이터 수집 및 전송 단말 장치의 데이터를 전송하는 경량 데이터 수집기 플랫폼
Logstash 	데이터 수집, 변환, 전송 데이터 가공 후 ElasticSearch에 저장가능한 동적 데이터 수집 파이프라인
Kibana 	데이터 시각화 확장형 사용자 인터페이스로 데이터를 구체적으로 시각화 가령 oracle db의 sqlplus와 흡사한 기능

Elastic Stack의 Flow

- Pipeline 구성이 가능



CRUD

데이터 저장 : POST

데이터 검색 : GET

데이터 수정 : PUT

데이터 삭제 : DELETE

데이터 전처리 & 전송

Mapping 설정

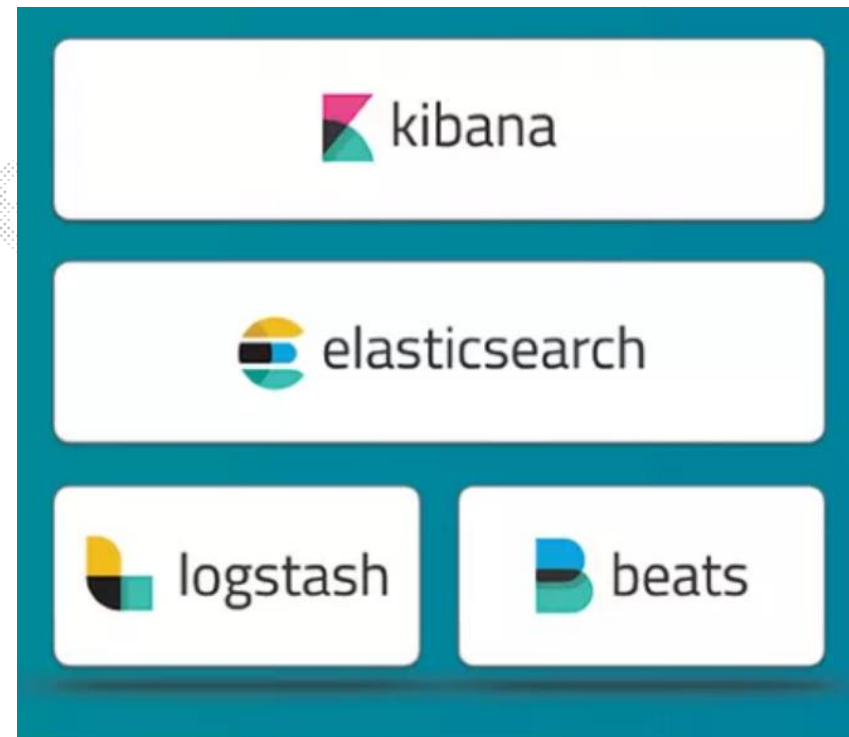
index 등록 / 데이터 시각화 / 대시보드 작성

- 루씬 기반의 Full Text로 검색이 가능한 오픈소스 텍스트 기반 분석용 검색 엔진
- **RESTful API**를 이용해 처리
 - HTML 기반의 RESTful를 활용하고 요청/응답 데이터 포맷으로 JSON 사용
 - 언어, 운영 체제, 시스템에 관계없이 다양한 플랫폼에서 활용 가능
- 대량의 데이터를 신속하고 실시간(Near real time)으로 저장, 검색 및 분석 할 수 있음
- **쿼리가 매우 빠르게 수행**
 - Rest API + JSON(데이터 구조)
- 속도와 확장성이 좋아 분산 구성이 가능하며, 분산 환경에서 데이터는 shard라는 단위로 나뉨

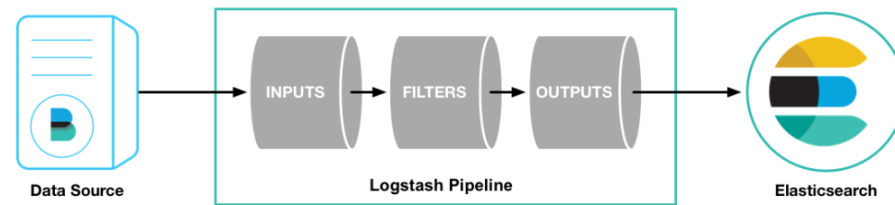
ElasticSearch 참고 및 주의 사항

- 완전 실시간은 아님
 - 색인된 데이터는 1초 뒤에나 검색이 가능
 - 내부적으로 commit과 flush같은 복잡한 과정을 거치기 때문
- Transaction Rollback을 지원하지 않음
 - 전체적인 클러스터의 성능 향상을 위해 시스템적으로 비용 소모가 큰 롤백과 트랜잭션을 지원하지 않음
- 데이터의 업데이트를 제공하지 않음
 - 업데이트 명령이 올 경우 기존 문서를 삭제하고 새로운 문서를 생성
 - 업데이트에 비해서 많은 비용이 들지만 이를 통해 불변성(Immutable)이라는 이점을 취함

- 확장형 사용자 인터페이스로 데이터를 구체적으로 시각화
- Elasticsearch 사용을 위한 Dev Tool 제공

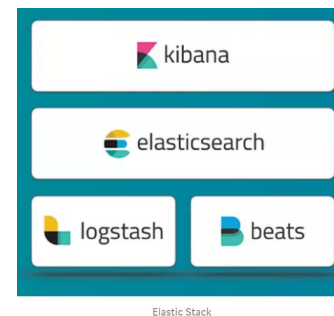


Elastic Stack

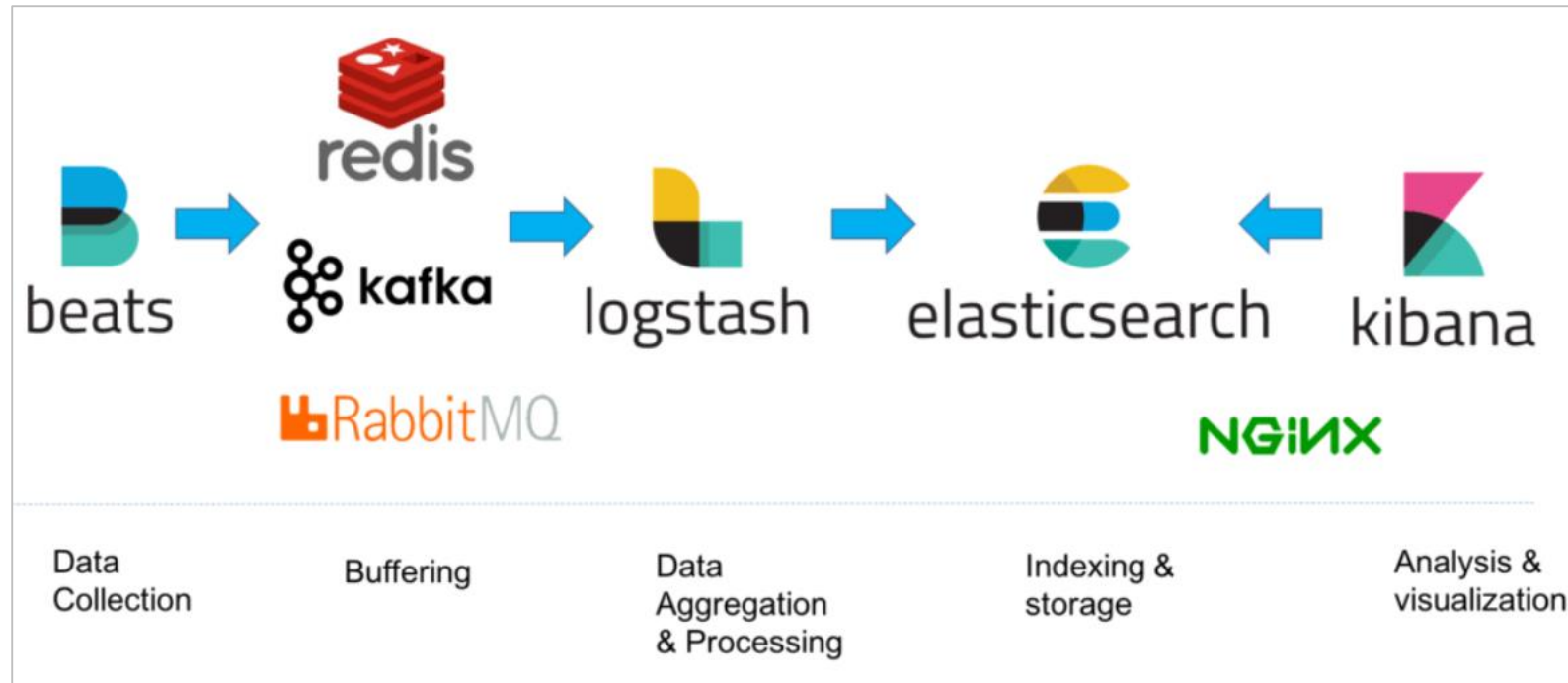


- 다양한 플러그인을 이용하여 데이터 집계 및 보관, 서버 데이터 처리
- 파이프라인으로 데이터를 수집하여 필터를 통해 변환 후 Elastic Search로 전송
 - 입력 : Beats, Cloudwatch, Eventlog 등의 다양한 입력을 지원하여 데이터 수집
 - 필터 :
 - 형식이나 복잡성에 상관없이 설정을 통해 데이터를 동적으로 변환
 - 수집된 데이터는 필터의 각 플러그인에 의해 가공
 - 가공 목적에 따라 여러 플러그인에서 사용 가능
 - 가령 grok 패턴을 사용해 비정형 데이터를 수집했다면 데이터 구조를 구성하거나 IP 주소로부터 지시적 위치 좌표를 생성하는 일이 가능
 - 출력 : Elastic Search, Email, ECS, Kafka 등 원하는 저장소에 데이터를 전송

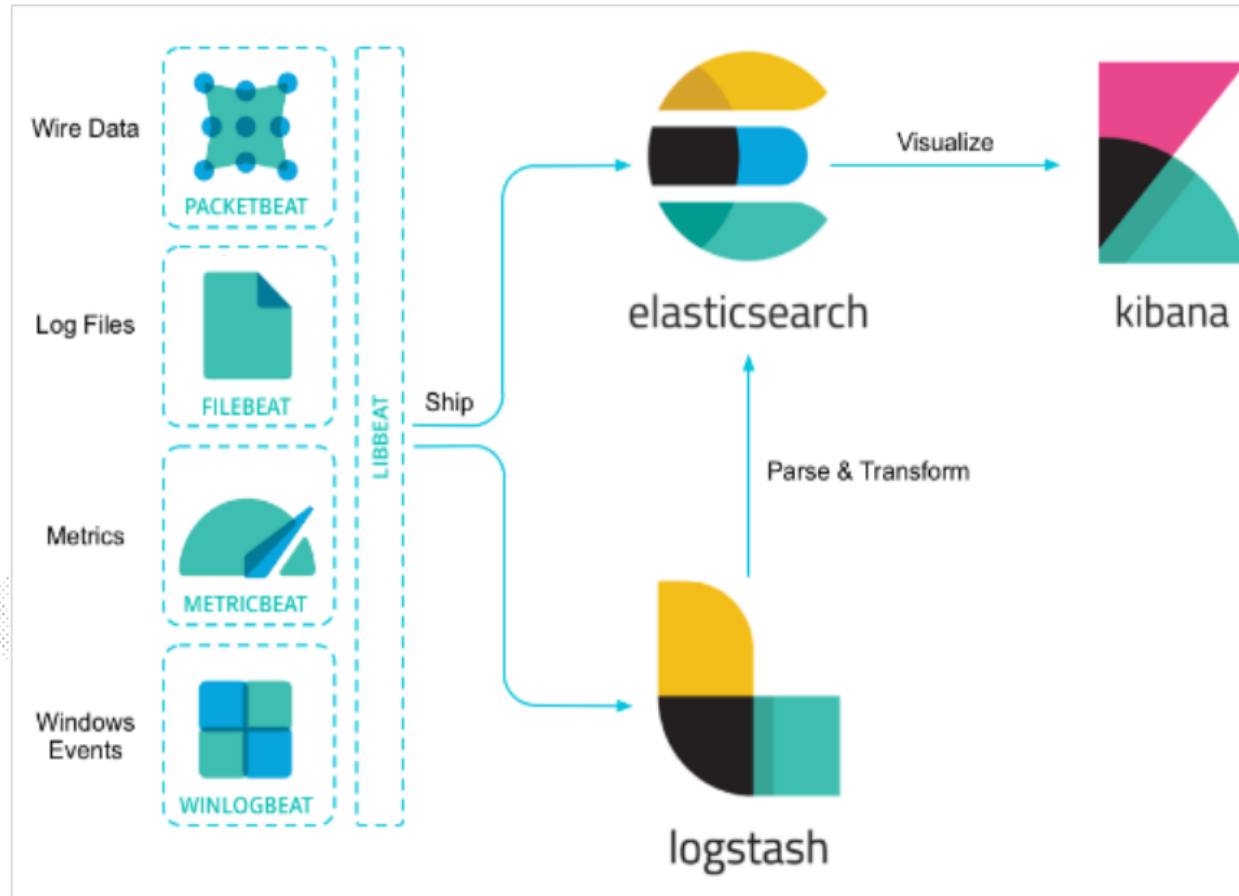
- 경량 에이전트로 설치되어 데이터를 Logstash 또는 Elastic Search로 전송하는 도구
- Logstash보다 경량화되어 있는 서비스
- Filebeat, Metricbeat, Packetbeat, Winlogbeat, Heartbeat 등이 있음
 - Packetbeat은 응용 프로그램 서버간에 교환되는 트랜잭션에 대한 정보를 제공하는 네트워크 패킷 분석기
 - **Filebeat는 서버에서 로그 파일을 제공**
 - Metricbeat은 서버에서 실행중인 운영 체제 및 서비스에서 메트릭을 주기적으로 수집하는 서버 모니터링 에이전트
 - Winlogbeat는 Windows 이벤트 로그를 제공



Elastic Stack의 Flow



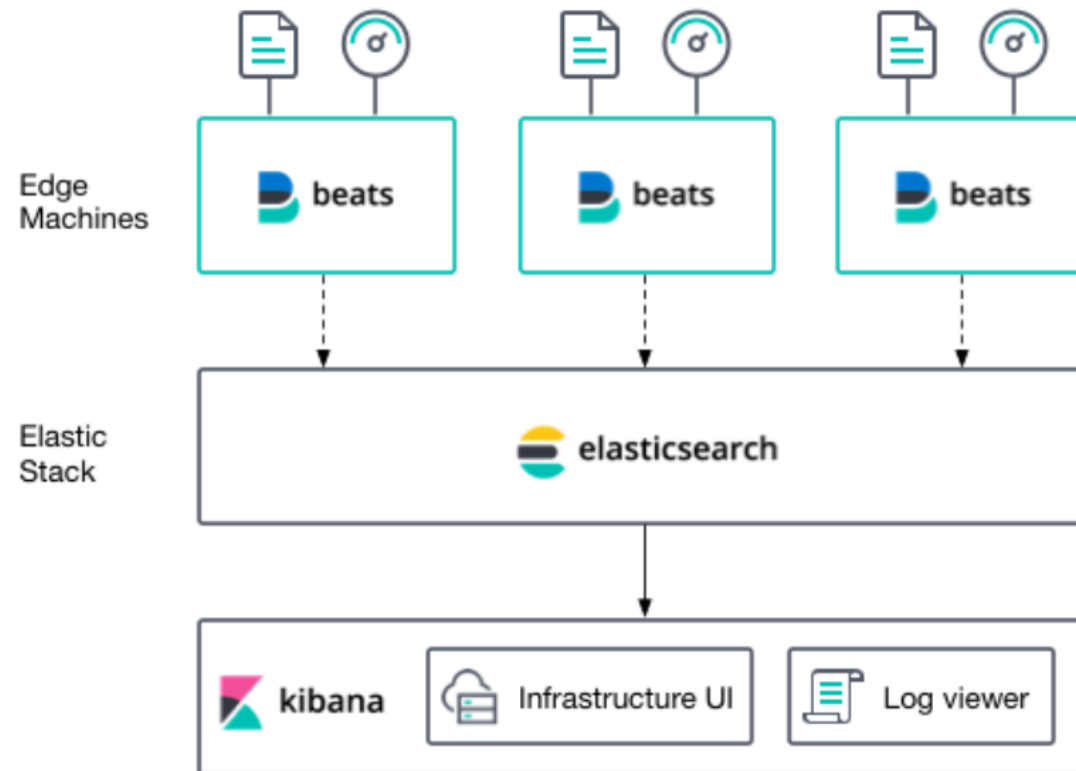
Elastic Stack의 Flow



Elastic Stack의 Flow

Elastic Stack의 Flow

Infrastructure monitoring components



ElasticSearch가 빠른 사유

- 역색인(inverted index) 개념이 도입되었기 때문
- 저장된 데이터 검색이 일반 DB보다 훨씬 빠르게 구성된 솔루션
 - 저장 구조 : JSON
- 예시 : oracle db의 용도
 - 데이터를 무결하게 영구적으로 저장해주는 솔루션
 - 역색인 기능은 일반 RDBMS엔 없음

index

- 색인
- 책 목차 개념

inverted index

- 역색인
- 책 맨 뒷부분의 키워드(term)로 검색하는 개념

데이터 색인이란? [용어 정리]

색인(indexing)

원본 문서를 검색이 가능한 구조인 토큰으로 변환하여 저장하는 일련의 과정

인덱스(index, indices)

색인 과정을 거친 결과 데이터가 저장되는 저장소,
Document들의 논리적인 집합 단위

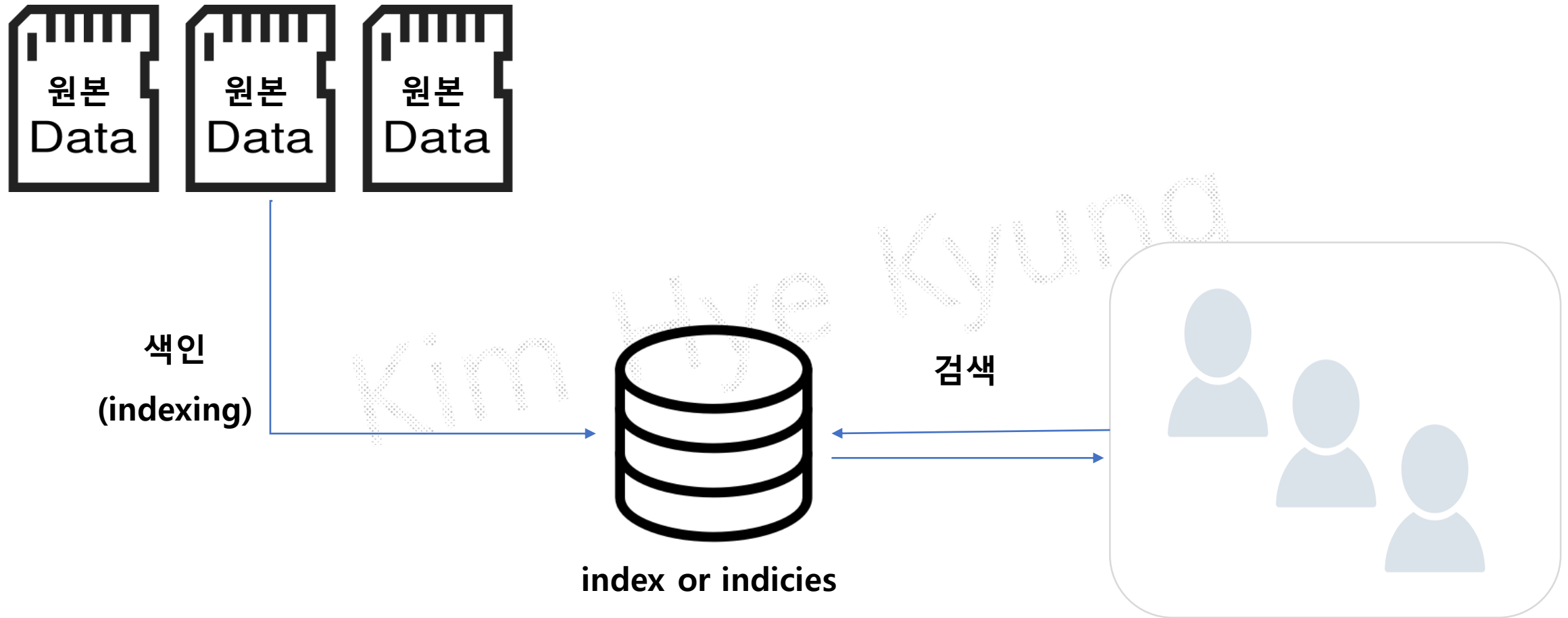
검색(search)

index 에 저장되어 있는 검색어 토큰들을 포함하고 있는 문서를 찾아가는 과정

질의

사용자가 원하는 document를 검색하거나 집계 결과를 도출하기 위해 입력하는 검색어 또는 조건

데이터 색인이란? [용어 정리]



역색인(inverted index) 이란?

- 역색인 파일을 만드는 것
 - 원문 자체를 변경한다는 의미는 아님
 - 색인 파일에 들어갈 토큰만 변경되어 저장되고 실제 문서의 내용은 변함없이 저장
 - 색인시 특정한 규칙과 흐름에 의해 텍스트를 변경하는 과정을 분석(Analyze)이라 함

역색인(inverted index) 이란?

- 종이책의 마지막 페이지에서 제공하는 색인 페이지와 흡사한 데이터 구조

- 예시

- '검색엔진' 이란 단어가 포함된 모든 문서를 찾아라?

- 일반적인 구조(RDBMS의 경우)

- 처음부터 끝까지 모든 문서를 읽어야만 원하는 결과를 얻을 수 있음

문서 번호	단어
1	엘라스틱서치
2	데이터베이스 역색인 검색엔진
3	문서
4	문서 검색엔진
5	데이터베이스

- 역색인 구조(Elasticsearch인 경우)

- 해당 단어만 찾으면 단어가 포함된 모든 문서의 위치를 알수 있음

- 빠른 검색이 가능

- 용어 : **Term**이란? 추출된 각 키워드 의미

- 데이터가 저장되었다 = indexing 되었습니다!!!

단어(Term)	문서 번호
엘라스틱서치	1
문서	3, 4
데이터베이스	2,5
역색인	2
검색엔진	2, 4

역색인 구조

- 모든 문서가 가지는 단어의 고유 단어 목록
- 해당 단어가 어떤 문서에 속해 있는지에 대한 정보
- 전체 문서에 각 단어가 몇 개 들어 있는지에 대한 정보
- 하나의 문서에 단어가 몇 번씩 출현했는지에 대한 빈도
- **데이터가 늘어날 경우**
 - 검색해야 할 행이 늘어나는 것이 아니라 역 인덱스가 가리키는 문서번호의 배열값이 추가
 - 따라서 큰 속도의 저하 없이 빠른 속도로 검색이 가능
 - 역 인덱스를 데이터가 저장되는 과정에서 만들기 때문에 데이터를 입력할 때 저장이 아닌 색인을 한다고 표현

단어(Term)	문서 번호
엘라스틱서치	1
문서	3, 4
데이터베이스	2,5
역색인	2
검색엔진	2, 4
백종원	10, 12

역색인 구조 – 예시를 통해 이해하기

- text 보유한 2개의 문서가 있는 경우

문서 1

The autumn sky is blue.

문서 2

The Autumn sky is high.

- 역색인 단계

1단계 : 문서를 토큰화(term으로 구성함을 의미)

2단계 : 토큰화된 단어에 대해 문서 상의 위치와 출현 빈도 등의 정보를 확인

역색인 구조 – 예시를 통해 이해하기

- 역색인 단계

1단계 : 문서를 토큰화된 정보

문서 1

The autumn sky is blue.

문서 2

The Autumn sky is high.

토큰
The
autumn
Autumn
sky
is
blue
high

역색인 구조 – 예시를 통해 이해하기

- 역색인 단계

2단계 : 토큰화된 단어에 대해 문서 상의 위치와 출현 빈도 등의 정보를 확인

문서 1

The autumn sky is blue.

문서 2

The Autumn sky is high.

토큰	문서 번호	Term의 위치	Term의 빈도
The	문서1, 문서2	1,1	2
autumn	문서1	2	1
Autumn	문서2	2	1
sky	문서1, 문서2	3,3	2
is	문서1, 문서2	4,4	2
blue	문서1	5	1
high	문서2	5	1

역색인 구조 – 예시를 통해 이해하기

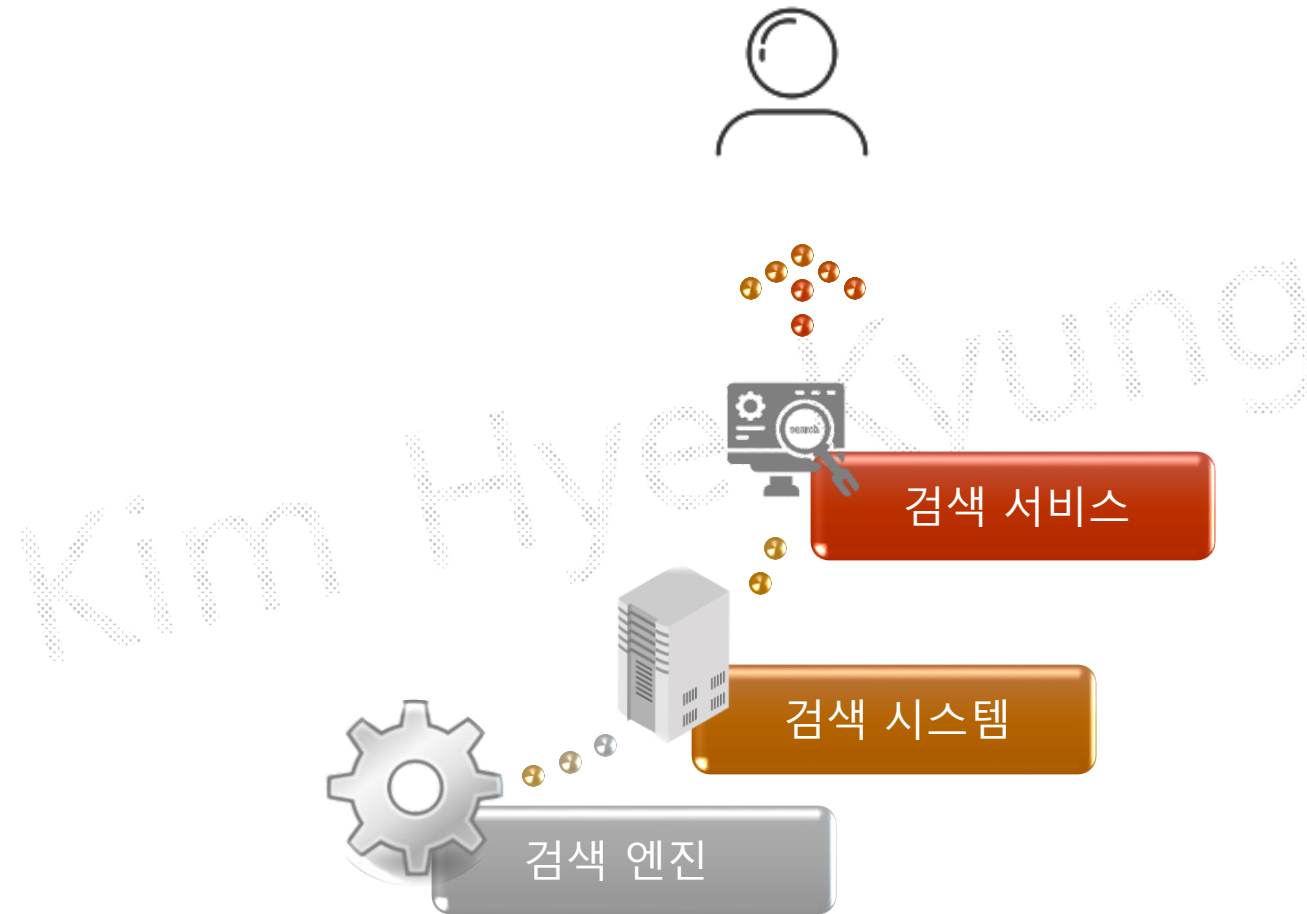
토큰	문서 번호	Term의 위치	Term의 빈도
The	문서1, 문서2	1,1	2
autumn	문서1	2	1
Autumn	문서2	2	1
sky	문서1, 문서2	3,3	2
is	문서1, 문서2	4,4	2
blue	문서1	5	1
high	문서 2	5	1

- 알수 있는 정보
 - 토큰이 어떤 문서의 어디에 위치
 - 몇번 나왔는지(빈도)
- 문제
 - **ElasticSearch는 대소문자를 구분!!**
 - autumn을 검색어로 지정할 경우?
 - 문서1, 문서2에서 검색되어야 함
 - 문제점
 - 정확하게 일치하는 데이터만 출력
 - 따라서 문서1만 검색됨
 - 해결책
 - 색인 전에 전체를 소문자로 변환한 다음 색인

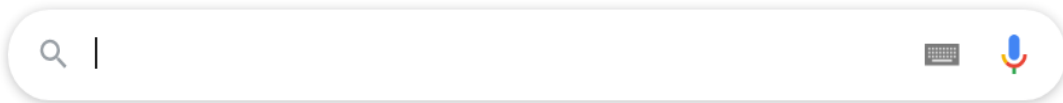
| 검색 시스템

검색 시스템 이해하기

- 검색 엔진?
- 검색 시스템?
- 검색 서비스?



검색 시스템 이해하기



- 구글 & 네이버의 대표 서비스
 - 검색 서비스
- 검색 시스템이란?
 - 사용자가 원하는 검색어에 대한 결과를 제공

검색 시스템 이해하기

검색 엔진(Search Engine)

- 광활한 웹에서 정보를 수집해 검색 결과를 제공하는 프로그램
- 검색 결과로 제공되는 데이터의 특성에 따라 구현 형태가 각각 달라짐
- 예
 - Yahoo
 - 검색 역사에 한 획을 그음
 - 디렉터리 기반의 검색 결과를 세계 최초로 제공
 - 영향력
 - 뉴스, 블로그, 카페 등 대범주에 따른 카테고리별 검색 결과를 대부분의 검색 업체에서 제공

검색 시스템 이해하기

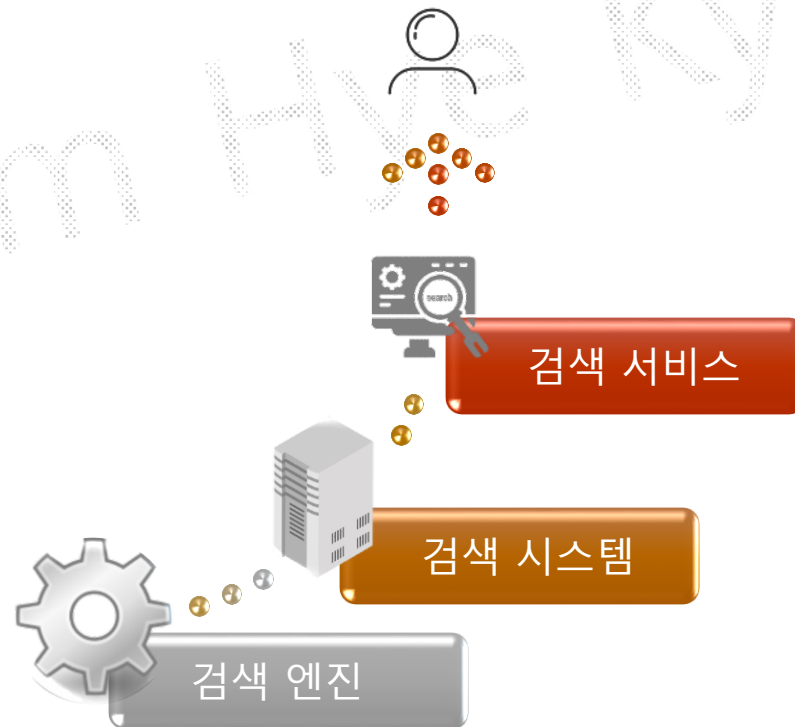
검색 시스템(Search System)

- 대용량 데이터를 기반으로 신뢰성 있는 검색 결과를 제공하기 위해 검색 엔진을 기반으로 구축된 시스템을 통칭하는 용어
- 수집기를 이용해 방대한 데이터 수집 -> 다수의 검색 엔진을 이용해 색인 -> 검색 결과를 UI로 제공
- 시스템 내부의 정책에 따라 가능한 작업
 - 관련도가 높은 문서를 검색 결과의 상위에 배치
 - 특정 필드나 문서에 가중치를 뒤서 검색의 정확도를 높일 수 있음

검색 시스템 이해하기

검색 서비스(Search Service)

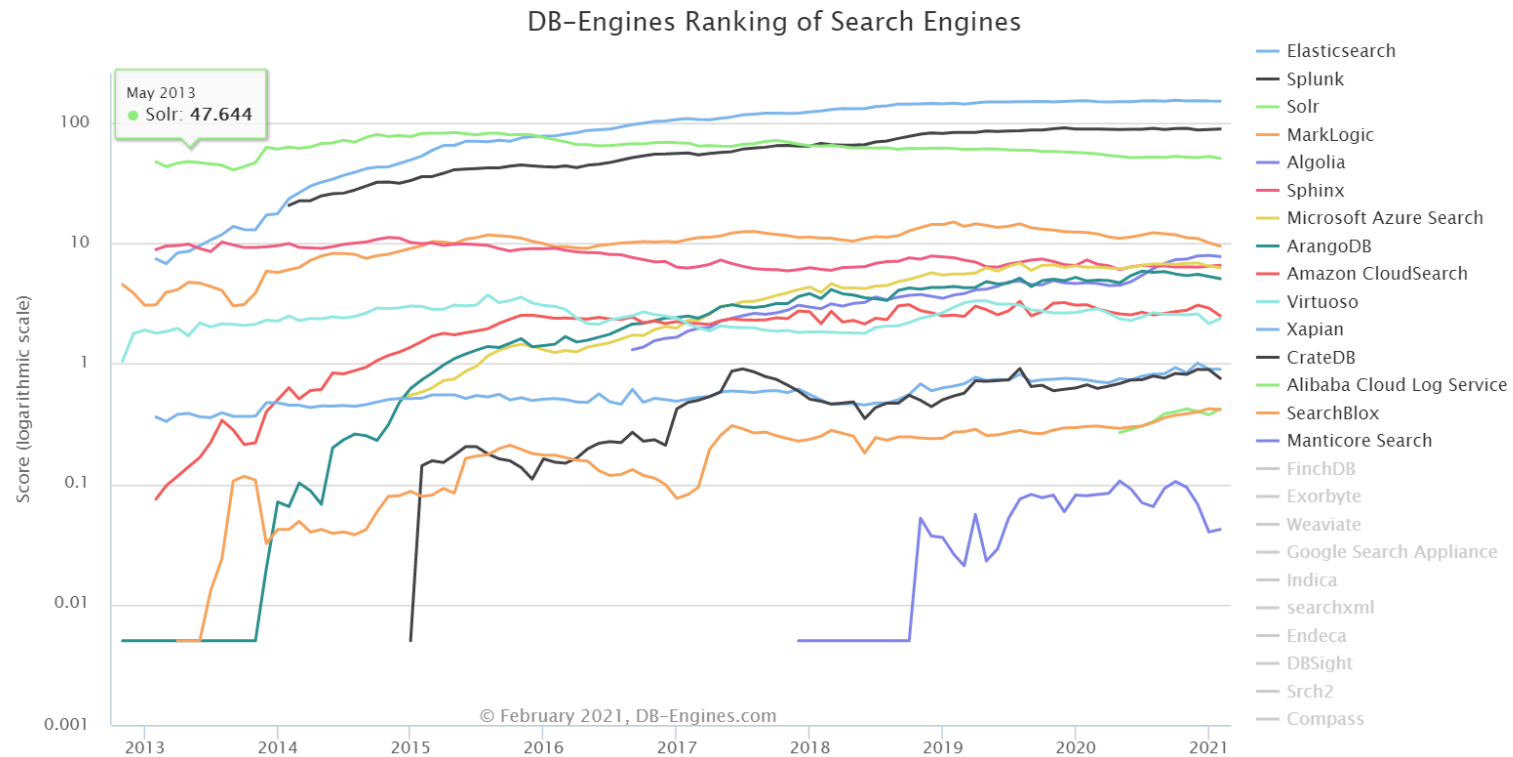
- 검색 엔진을 기반으로 구축한 검색 시스템을 활용해 검색 결과를 서비스로 제공



검색 시스템 트렌드

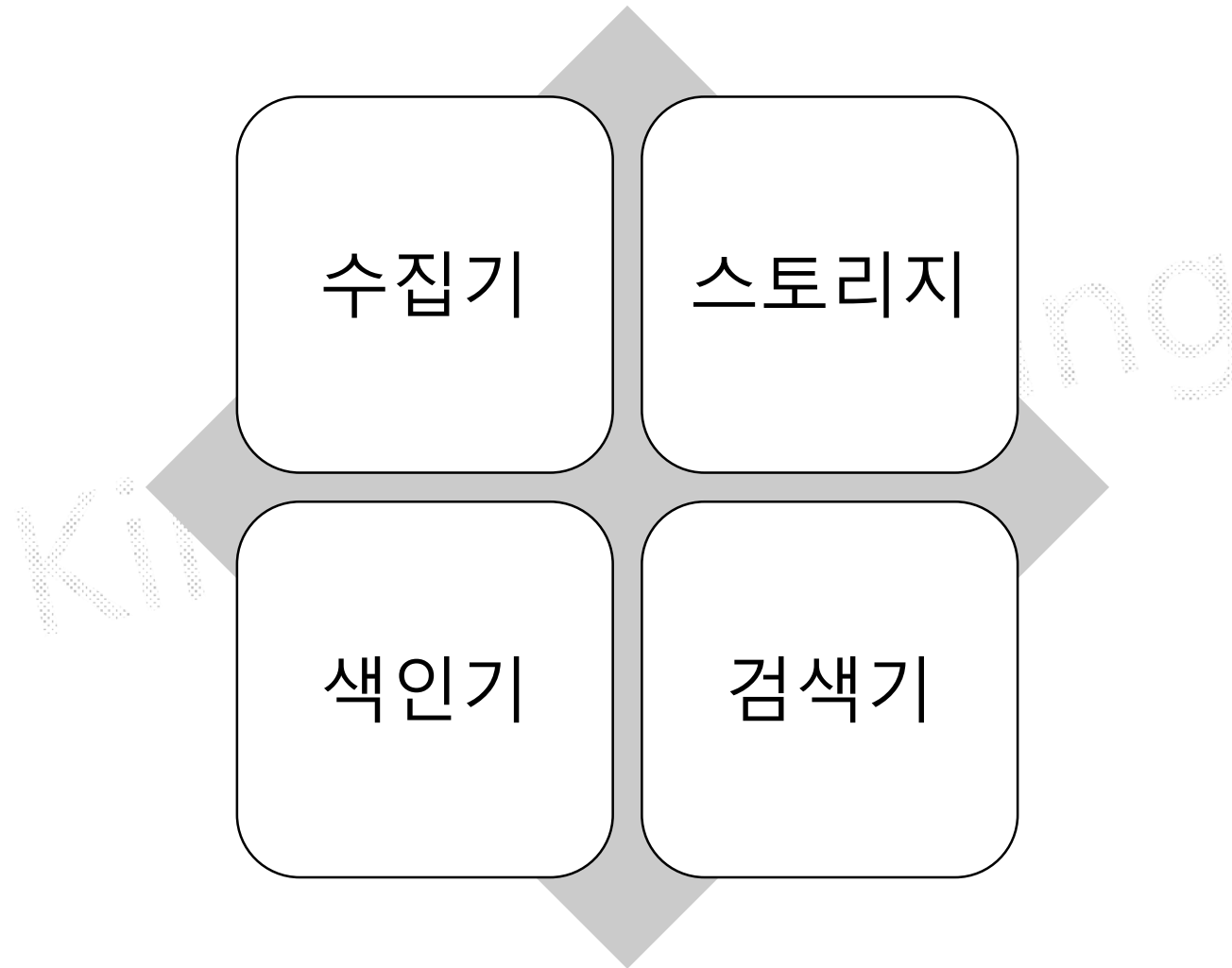
검색엔진 순위 트렌드 : 2016년 무렵 부터 ES가 가장 많이 사용됨

https://db-engines.com/en/ranking_trend/search+engine



Click at a system in the legend
to hide or show its trend line

검색 시스템 구성 요소



검색 시스템 구성 요소

- 검색 시스템의 구성



검색 시스템 구성 요소

수집기

- 정보 수집

스토리지

- 수집한 데이터를 저장

색인기

- 수집한 데이터를 검색에 적절한 형태로 변환

검색기

- 색인된 데이터에서 일치하는 문서를 찾는 검색기

검색 시스템 구성 요소

수집기

- 정보 수집
- 웹사이트, 블로그, 카페 등 웹에서 필요한 정보를 수집하는 프로그램
- 크롤러(Crawler), 스파이더(Spider), 웜(Worms), 웹로봇(Web Robot)등으로 불리
- 수집 대상 : 파일, 데이터베이스, 웹페이지 등 웹상의 대부분의 정보가 수집 대상
- 파일의 경우 : 수집기가 파일명, 파일 내용, 파일 경로 등의 정보를 수집하고 저장하면 검색 엔진이 저장된 정보를 검색하고, 사용자 질의에 답함

스토리지

- 수집한 데이터를 저장
- 데이터베이스에서 데이터를 저장하는 물리적인 저장소
- 검색 엔진은 색인한 데이터를 스토리지에 보관

검색 시스템 구성 요소

색인기

- 수집한 데이터를 검색에 적절한 형태로 변환
 - 검색 엔진이 수집한 정보에서 사용자 질의와 일치하는 정보를 찾으려면 수집된 데이터를 검색 가능한 구조로 가공 및 저장해야 함
- 다양한 형태소 분석기를 조합해 정보에서 의미 있는 용어를 추출 및 검색에 유리한 역색인 구조로 데이터 저장

검색기

- 색인된 데이터에서 일치하는 문서를 찾는 검색기
- 사용자 질의를 입력받아 색인기에서 저장한 역색인 구조에서 일치하는 문서를 찾아 결과로 반환
- 질의와 문서가 일치 하는지는 유사도 기반의 검색 순위 알고리즘으로 판단
- 색인기와 마찬가지로 형태소 분석기를 이용해 사용자 질의에서 유의미한 용어를 추출해 검색
- 사용하는 형태로 분석기에 따라 검색 품질이 달라짐



쉬어가기 – 용어 이해하기

- index
 - RDB에서의 index
 - where 절의 쿼리와 join을 빠르게 만들기 위한 보조 데이터 도구
 - Pk로 설정시에는 자동으로 index 적용
 - Pk 는 데이터 구분을 위한 검색 기준 데이터
 - ES에서의 index
 - **데이터를 저장하는 행위를 의미**
 - 내부적으로 데이터를 term 으로 구분 후에 역색인 수행해서 색인 구조의 메타정보 생성

관계형 데이터베이스와의 차이점 이해하기

관계형 데이터베이스

- 데이터베이스는 데이터를 통합 관리하는 데이터의 집합
- 저장되는 구조
 - 중복을 제거
 - **정형 데이터** 구조화
 - 행과 열로 구성
 - 테이블 구조로 저장
- sql 문을 이용해 원하는 정보의 검색이 가능
- 텍스트 매칭을 통한 단순한 검색만 가능
- 텍스트를 여러 단어로 변형하거나 여러 개의 동의어나 유의어를 활용한 검색은 불가능

검색 엔진

- 데이터베이스에서는 불가능한 **비정형 데이터**를 색인하고 검색 할 수 있음
- 형태소 분석을 통해 사람이 구사하는 자연어 처리가 가능해짐
- 역색인 구조를 바탕으로 빠른 검색 속도 보장

관계형 데이터베이스와의 차이점 이해하기

- CRUD 작업의 차이점
 - ElasticSearch
 - HTTP를 통해 JSON 형식의 RESTful API 사용

Elastic Search	Relation DB	CRUD 기능
GET	데이터 조회(select)	데이터 조회
PUT	데이터 생성, 수정(insert, update)	데이터 생성, 수정
POST	데이터 생성, 수정, 검색(update, select)	인덱스 생성, 수정, 조회
DELETE	데이터 삭제(delete)	데이터 삭제
HEAD	-	인덱스 정보 확인

기본 명령어

Elastic Search	Relation DB	CRUD
GET	Select	Read
PUT	Update, Insert	Create, Update, Insert
POST	Insert, Update	Create, Read, Update
DELETE	Delete	Delete

REST API[Representational State Transfer]

- 직관적인 의미 : 대표적인 상태 전달

REST란?

웹에 존재하는 모든 자원(이미지, 동영상, DB 자원)에 고유한 URI를 부여해 활용하는 것으로,
자원을 정의하고 자원에 대한 주소를 지정하는 방법론을 의미

Restful API란?

REST 특징을 지키면서 API를 제공하는 것을 의미

HTTP 헤더(header)와 URL만 사용해 다양한 형태의 요청을 할 수 있는 HTTP 프로토콜을

최대한 활용하도록 고안된 아키텍처

ElasticSearch의 RESTful API

- API 요청 구조

<http://host:port/인덱스/타입/문서id>

- RDBMS의 sql과 비교

- 예시 : customer의 주소(address)에 "서초"가 포함된 고객의 모든 데이터 검색을 요할 경우

- RDBMS

select * from customer from address like '%서초%' -> table 형태로 검색

- ElasticSearch

GET http://ip:port/customer/_search?q=address:서초 -> json 형태로 검색

| 검색 시스템 & Elasticsearch

ElasticSearch 특징

- Shay Banon이 Lucene을 바탕으로 개발한 오픈소스 분산 검색엔진
- 2010년 2월 첫 버전이 공개
- 설치와 서버 확장이 매우 편리
- 실시간 검색 (Near Real Time)
 - 검색 엔진이 내장되어 있기 때문
- 자체적으로 클러스터를 탐색하고 관리

검색 엔진 관점에서의 Elasticsearch 특징

- 대용량 데이터를 빠르게 검색하기 위해 **NoSQL(No Structured Query Language)** 많이 사용
- 분류가 가능하고 분산 처리를 통해 실시간에 준하는 빠른 검색이 가능
- 기존 데이터베이스로 처리하기 힘겨운 대량의 비정형 데이터 검색 가능
- 전문 검색(full text)과 구조 검색 모두 지원
 - 전문 검색이란?
 - 내용 전체를 색인해서 특정 단어가 포함된 문서를 검색하는 것 의미
 - RDB – 전문 검색에 부적합
 - Elasticsearch – 다양한 기능별, 언어별 플러그인을 조합해 빠른 검색이 가능
- MongoDB나 Hbase처럼 대용량 스토리지로도 활용 가능

검색 엔진 관점에서의 Elasticsearch 특징

- **JSON 기반의 스키마 없는 저장소**

- 검색 엔진이지만, NoSQL처럼 사용할 수 있음
- 데이터 모델을 JSON으로 사용하고 있어서, 요청과 응답을 모두 JSON 문서로 주고받음
- 소스 저장도 JSON 형태로 저장
- 스키마를 미리 정의하지 않아도, JSON 문서를 넘겨주면 자동으로 인덱싱
- 숫자나 날짜 등의 타입은 자동으로 매핑
- 구조화된 JSON으로 검색 자체적으로 통계 정보 반환

검색 엔진 관점에서의 Elasticsearch 특징

- 통계 분석
 - 비정형 로그 데이터 수집 후 통계 분석 가능
 - 키바나 연계시 실시간으로 로그 데이터 시각화 가능
- 스키마리스
 - RDB와 달리 다양한 형태의 문서도 자동으로 색인 및 검색 가능
- RESTful API
 - HTTP 기반의 RESTful API 로 요청 및 JSON 형태의 응답으로 개발언어, 운영체제, 시스템에 비종속적
 - 이기종 플랫폼에서 이용 가능

검색 엔진 관점에서의 Elasticsearch 특징

- Document-Oriented
 - 여러 계층의 데이터를 JSON 형식의 구조화된 문서로 인덱스에 저장 가능
 - 계층 구조로 문서도 한번의 쿼리로 쉽게 조회 가능
- 역색인(inverted index)
 - NoSQL지원하는 MongoDB or Cassandra와 달리 역색인 지원
- 확장성과 가용성
 - 대용량 데이터 저장 및 색인시 막대한 비용과 시간 소비되나 ElasticSearch를 분산 구성해서 확장한다면 대량의 문서를 좀 더 효율적으로 처리 가능
 - 분산 환경에서 샤드(shard)라는 작은 단위로 나뉘어 제공, 인덱스를 만들 때마다 샤드의 수를 조절해서 데이터의 종류와 성격에 따라 분산해서 빠르게 처리 가능

검색 엔진 관점에서의 Elasticsearch 특징

- Multi-tenancy
 - 서로 상이한 인덱스일지라도 검색할 필드명만 동일하다면 여러 개의 인덱스를 한번에 조회 가능
 - 하나의 elasticsearch 서버에 여러 인덱스를 저장하고, 여러 인덱스의 데이터를 하나의 쿼리로 검색할 수 있음

예제 1 Multi-tenancy 예제 쿼리

```
# log-2012-12-26 인덱스에 로그 저장
curl -XPUT http://localhost:9200/log-2012-12-26/hadoop/1 -d '{
  "projectName" : "hadoop",
  "logType": "hadoop-log",
  "logSource": "namenode",
  "logTime": "2012-12-26T14:12:12",
  "host": "host1",
  "body": "org.apache.hadoop.hdfs.StateChange: DIR* NameSystem.completeFile"
}'

# log-2012-12-27 인덱스에 로그 저장
curl -XPUT http://localhost:9200/log-2012-12-27/hadoop/1 -d '{
  "projectName" : "hadoop",
  "logType": "hadoop-log",
  "logSource": "namenode",
  "logTime": "2012-12-27T02:02:02",
  "host": "host2",
  "body": "org.apache.hadoop.hdfs.server.namenode.FSNamesystem"
}'

# log-2012-12-26, log-2012-12-27 인덱스에 한번에 검색 요청
curl -XGET http://localhost:9200/log-2012-12-26, log-2012-12-27/_search
```


ElasticSearch의 query 한글 문서

- <https://www.elastic.co/guide/kr/elasticsearch/reference/current/gs-executing-aggregations.html>

Kim Hye Kyung

| Elasticsearch

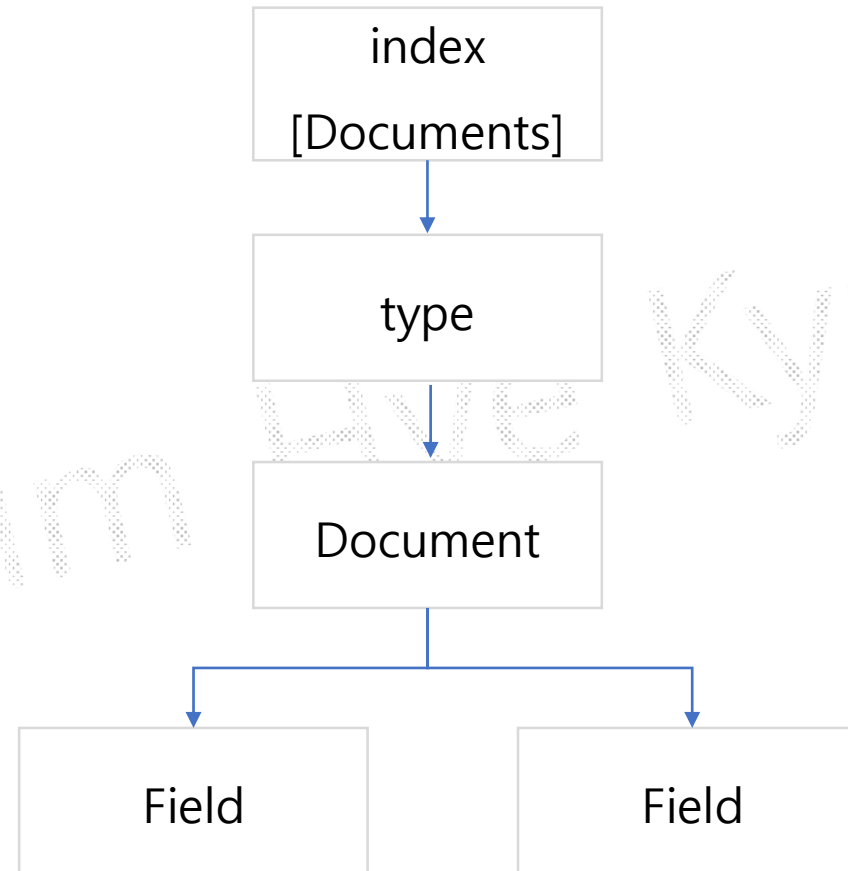
Elasticsearch 구성 이해하기

- 관계형 데이터베이스와 elasticsearch 용어 비교

관계형 데이터베이스	elasticsearch	
Database	Index	
Table	Type	
Row	Document	<code>_doc</code>
Column	Field	<code>{ "field" : "value" }</code>
Schema 구조와 제약조건	Mapping	Json 포맷에 적합한 구조
Index 빠른 검색이 가능한 옵션	Everything is indexed	데이터가 저장될 의미 : 역색인 관리등 포함
SQL	Query DSL	

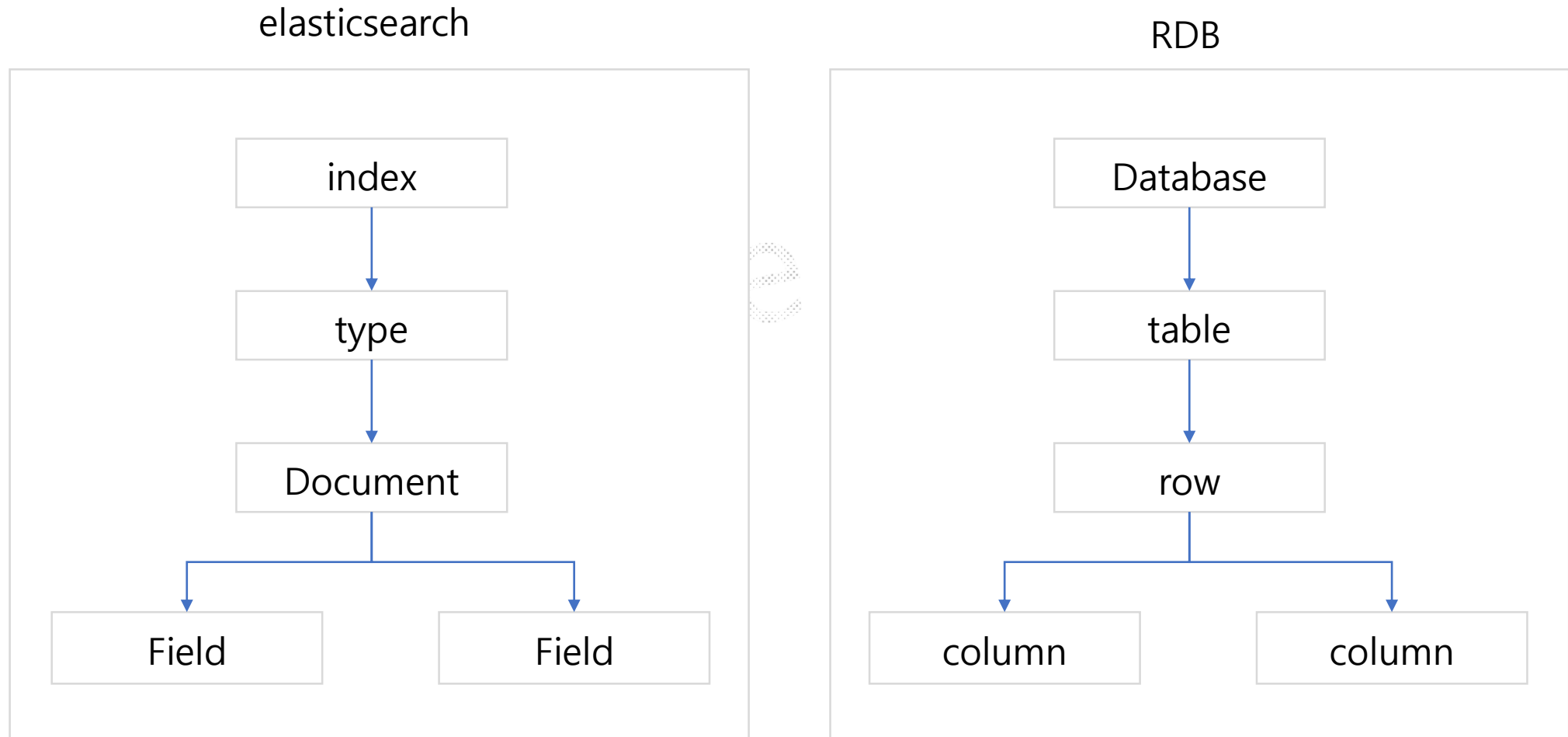
Elasticsearch 구성 이해하기

- 구성

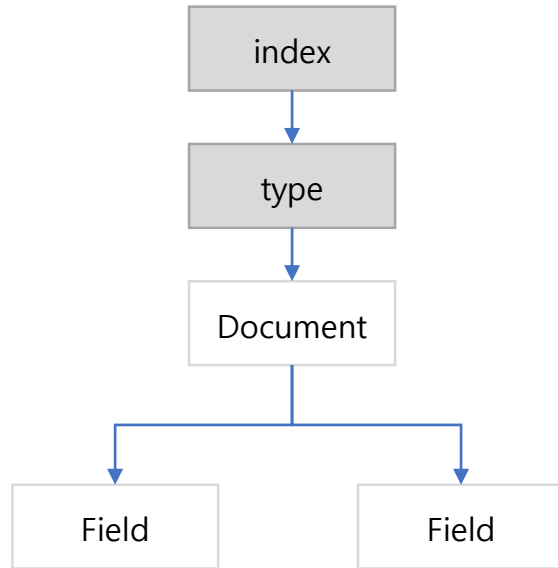


Elasticsearch 구성 이해하기

- RDB와의 비교

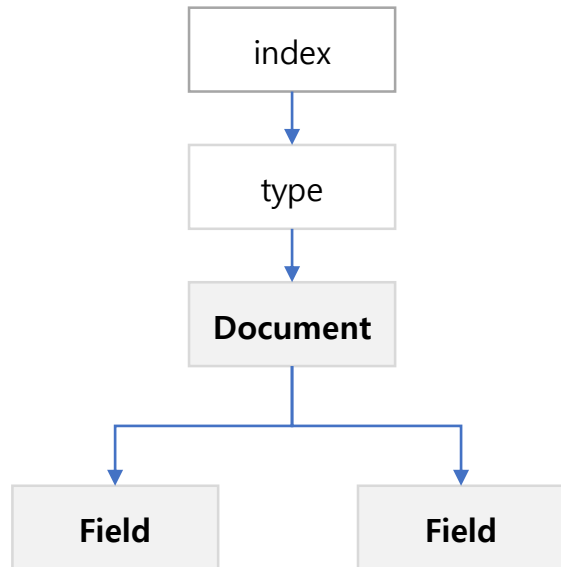


Elasticsearch 구성 이해하기



용 어	설 명
index	<p>데이터 저장 공간</p> <p>이름은 소문자 여야만 함</p> <p>추가, 수정, 삭제, 검색은 RESTFul API로 수행</p>
shard	<p>인덱스 내부에 색인된 데이터는 물리적인 공간에 여러 개의 파티션으로 나뉘어 구성</p> <p>이 파티션을 샤드라고 함</p> <p>다수의 샤드로 문서를 분산 저장하고 있기 때문에 데이터 손실 위험 최소화</p> <p>하나의 엘라스틱서치에는 2개의 물리적인 노드 존재</p> <p>인덱스 문서 조회시 2개의 노드 모두 조회하고 각 데이터를 취합한 후 하나로 합쳐서 제공</p>
type	<p>인덱스의 논리적 구조, RDB의 table로 간주</p> <p>인덱스 속성에 따라 분류</p> <p>v6.1 이상 부터는 인덱스당 하나의 타입만 사용</p>

Elasticsearch 구성 이해하기



용 어	설 명
문서	하나의 행을 문서라 부름(DB row와 흡사한 개념) 데이터가 저장되는 최소 단위 JSON 포맷으로 데이터가 저장 다수의 field로 구성
필드	문서를 구성하기 위한 속성 데이터의 형태에 따라 용도에 맞는 데이터 타입을 정의해야 함 RDBMS의 column과 흡사한 개념
매핑	필드의 구조와 제약조건에 대한 명세 RDB의 개념 관점에서 스키마라 함 문서의 필드와 필드의 속성을 정의하고 그에 따른 색인 방법을 정의하는 프로세스

Elasticsearch 구성 이해하기

- 문서를 색인화 한다는 의미

- Rest API를 통해 index에 document를 추가

- Rest API로 document 추가 및 조회

- -d 옵션
 - 추가할 데이터를 json 포맷으로 전달
 - -H 옵션
 - 헤더를 명시
 - json으로 전달하기 위해서 application/json으로 작성
 - ?pretty
 - 결과를 예쁘게 보여주도록 요청

ver. window

```
C:\Users\Kimhyekyung>curl -H "Content-Type:application/json" -XPUT "localhost:9200/user/_doc/1?pretty" -d '{"name":"khk", "age":20}'
{
  "_index" : "user",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

```
curl -XPUT 'localhost:9200/user/1?pretty' -d '{"name" : "khk", "age" : 20}' -H 'Content-Type: application/json'
```


Elasticsearch 구성 이해하기

- Rest API로 document 추가 및 조회
 - 조회

```
curl -XGET "localhost:9200/user/_doc/1?pretty"
```

```
C:\Users\Kimhyekyung>curl -XGET "localhost:9200/user/_doc/1?pretty"
{
  "_index" : "user",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 1,
  "_seq_no" : 0,
  "_primary_term" : 1,
  "found" : true,
  "_source" : {
    "name" : "khk",
    "age" : 20
  }
}
```

|| 용어 이해하기

용어

- Cluster
- ScaleUp/ScaleOut
- Node
- Index
- Document
- Shards
- Replicas

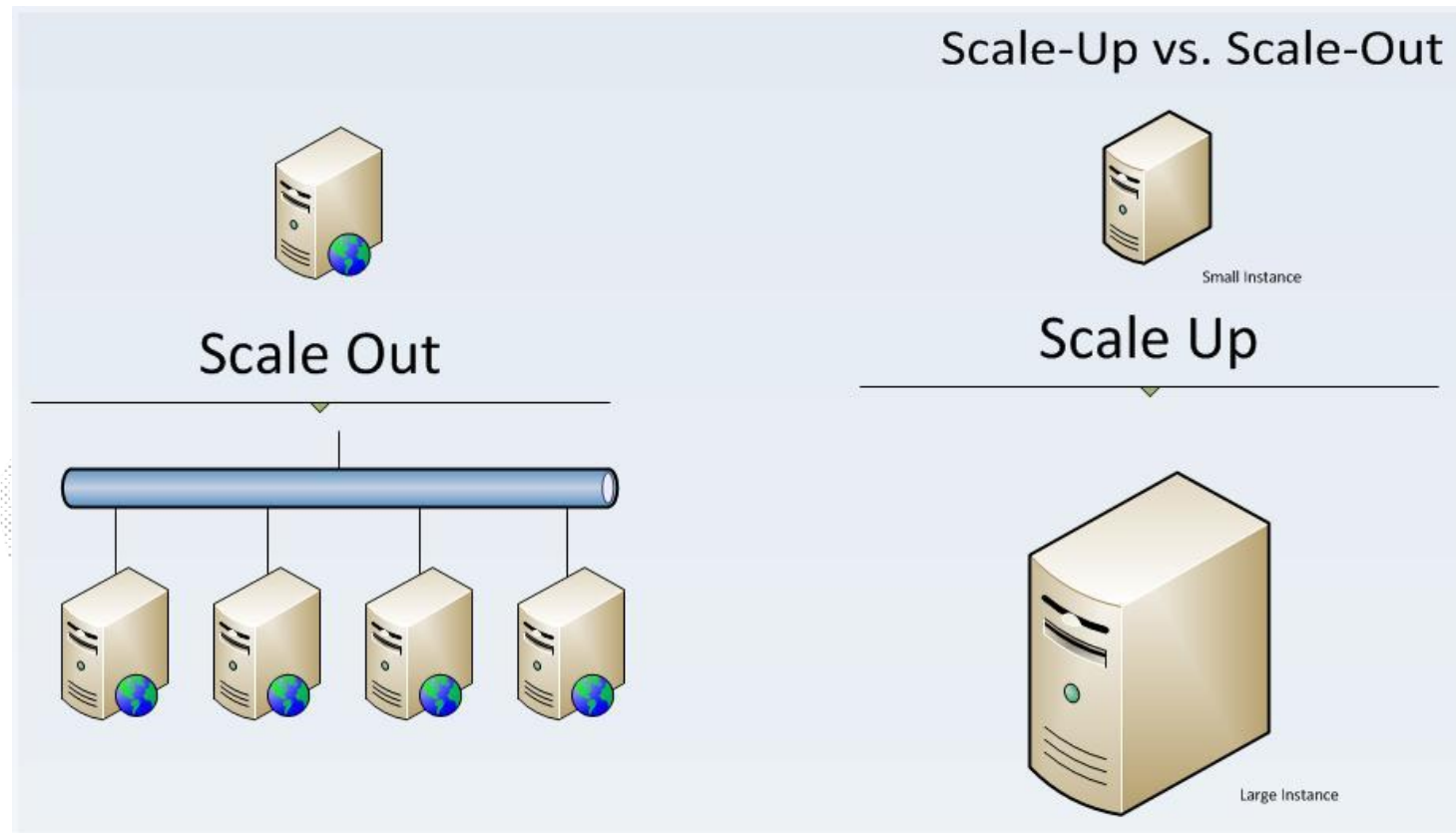


용어 : Cluster[클러스터)란?

- 컴퓨터의 집합을 의미
 - 모든 데이터를 함께 가지고 있는 한개 또는 그 이상의 노드의 집합
- 여러 대의 일반 워크스테이션(노드)을 네트워크로 연결하여 하나의 PC 처럼 작동하게 하는 기술
- 워크스테이션의 CPU 성능이 좋아지고 네트워크 속도 또한 엄청나게 발달하여 클러스터의 실제 적용이 가능해짐
- 클러스터 PC들의 OS는 오픈 소스로 인해 자유롭게 튜닝이 가능한 리눅스 사용
- 컴퓨팅 파워를 증가시키키 위한 방법

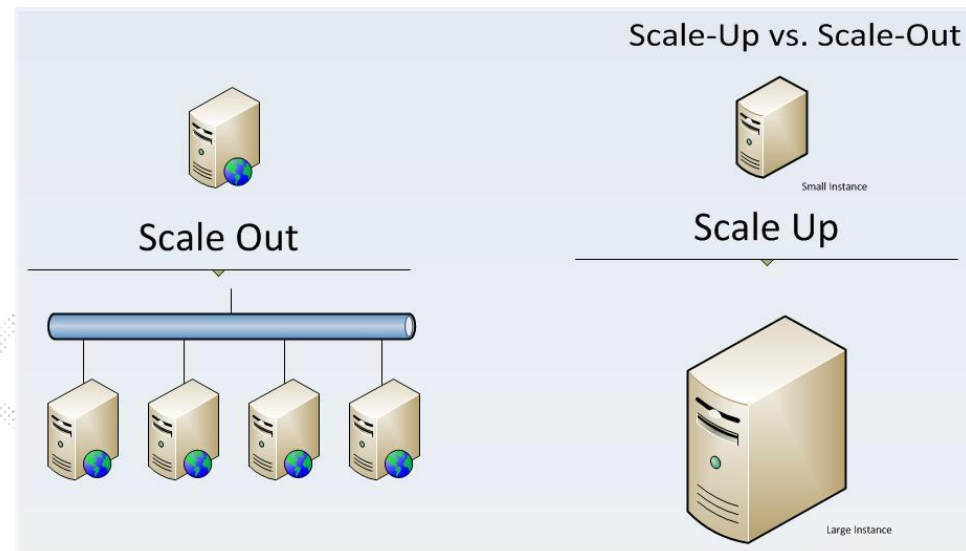
용어 : ScaleUp/ScaleOut

- Scale up : 사양추가
- Scale out : 장비추가



용어 : ScaleUp/ScaleOut

- Scale up : 사양추가
 - 수직 확장 개념
 - 복잡한 계산이 많을 경우 사양(CPU, RAM, DISK 구성등) 고가의 장비로 대체하여 처리 속도 향상
- Scale out : 장비추가
 - 수평 확장 개념
 - 분산처리, 병렬처리등으로 불리우기도 함
 - 단순 데이터 처리가 많은 환경에 적합
 - 단, 병렬 구조된 서버들에 데이터를 어떻게 동기화 해야 하며 세션은 어떤식으로 공유 해야 할지에 대한 기술적인 한계가 있음
 - 데이터의 정합성(데이터가 서로 모순 없이 일관되게 일치해야 하는 경우) 유지에 대한 요건이 별로 어렵지 않을 경우 적합



용어 : Scale out과 Scale up 비교

	Scale out	Scale up
확장성	하나의 장비에서 처리하던 일을 여러 장비로 나눠서 처리(수평확장) 지속적 확장이 가능	더 빠른 속도의 CPU로 변경 또는 더 많은 RAM 추가 등의 HW 장비 성능 향상(수직 확장) 성능 확장에 한계가 있음
서버 비용	비교적 저렴한 서버 사용 일반적으로 비용 부담이 적음	성능 증가에 따른 비용 증가폭이 큼 일반적으로 비용 부담이 큼
운영 비용	대수가 늘어날수록 관리 편의성이 떨어짐 서버의 사용 비용을 포함한 운영 비용 증가	관리 편의성, 운영 비용은 스케일 업에 따라 큰 변화 없음
장애	읽기/쓰기가 여러대의 서버에 분산되어 처리됨으로 장애 시 전면 장애의 가능성이 적음	한대의 서버에 부하가 집중 장애시 장애 영향도가 큼
주요 기술	Sharding, NoSQL, In Memory Cache등	고성능 CPU, Memory 확장, SSD
주요 용도 장/ 단점	분산처리 시스템, 점진적 증가 가능, Scale up 보다 저렴 단점 : 설계, 구축, 관리 비용 증가	고성능 Legacy Application 구축이 쉽고 관리 용이 단점 : 단계적 증가가 어렵고 근본적인 해결이 안 될수도 있음

용어 : Node

- Cluster의 일부로 단일 서버 의미
- 데이터를 보관하고 Cluster indexing과 검색 능력에 관여

Kim Hye Kyung

용어 : Index

- 비슷한 특성을 가진 document의 집합
- 이름으로 구분
- 데이터 저장 공간
- 하나의 물리 노드에 여러개 논리 인덱스 생성
- 하나의 인덱스가 여러 노드에 분산 저장 (M:N)

용어 : Document

- indexing 될수 있는 정보의 기본 단위
- 인터넷 데이터 교환 포맷인 JSON으로 표현됨
- 데이터가 저장되는 최소 단위
- JSON 포맷으로 저장
- DB의 Row에 대응됨

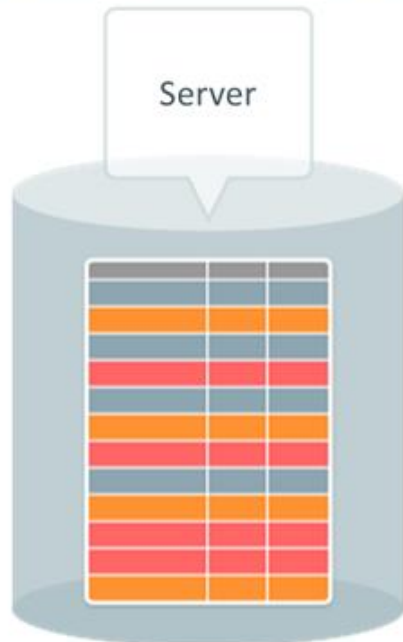
용어 : Sharding & Shard

- 단일의 데이터를 다수의 데이터베이스로 쪼개서 나누는 것 의미
- 데이터가 클때, 안정적으로 보관시 사용
- 데이터가 급격히 증가하게 되거나 트래픽이 특정 DB로 몰리는 상황을 대비해, 빠르고 유연하고 안전한 DB증설이 필요
- 필요한 데이터만 빠르게 조회할 수 있기 때문에 쿼리 자체가 가벼움
- 샤드란?
 - 샤딩을 통해 나누어진 블록들의 구간을 의미
 - Elasticsearch는 분산 데이터베이스(분산 검색엔진)이기때문에 이렇게 데이터를 나누어 저장하는 방식으로 대용량 데이터에 대한 분산처리를 가능하게 함

용어 : Sharding & Shard

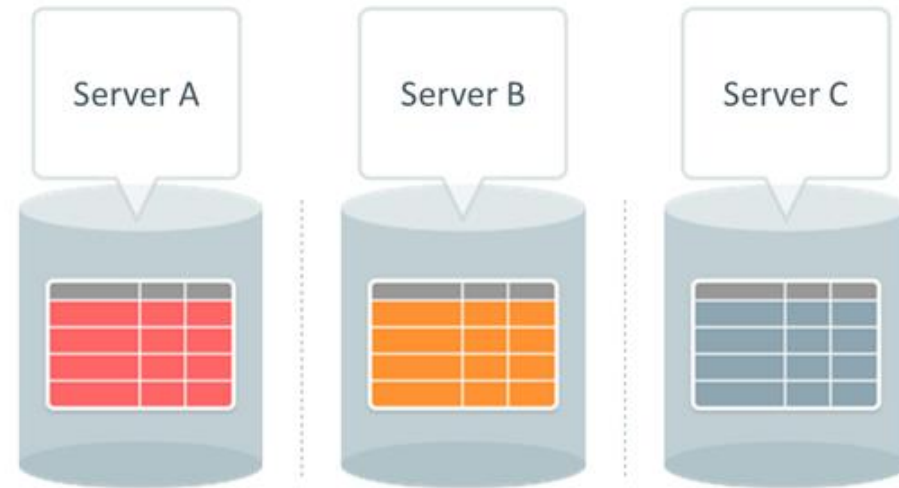
Single Database VS. Elastic Sharding

Data partitioned into smaller pieces for performance, availability and manageability



Unsharded table in 1 database

Extends partitioning to distributed databases for scalability and fault isolation



Sharded table in 3 databases

관계형 데이터베이스와의 차이점

샤딩(sharding)

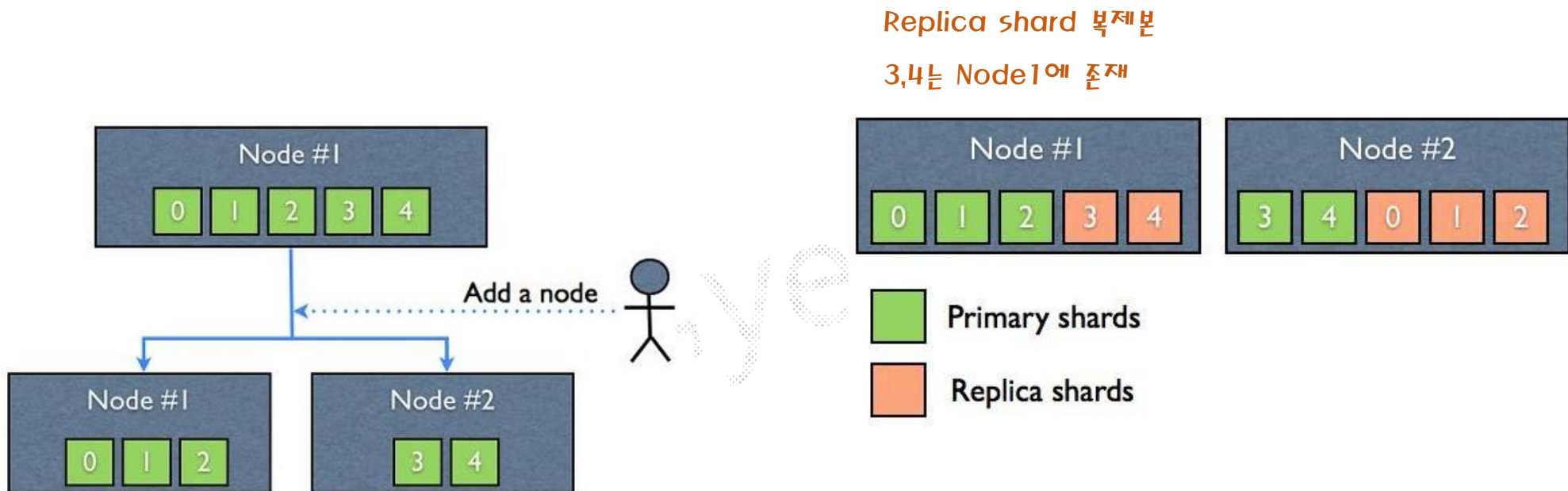
- 데이터를 분산해서 여러 그룹으로 쪼개어 이 그룹들을 각기 다른 머신에 저장하는 처리 방법
- 파티셔닝(Partitioning)이라고도 함
 - Elasticsearch에서 Scale Out을 위해 index를 여러 shard로 쪼갬 것
- 기본적으로 1개가 존재하며, 검색 성능 향상을 위해 클러스터의 샤드 갯수를 조정하는 튜닝을 하기도 함

replica

- 또 다른 형태의 shard
- 노드를 손실했을 경우 데이터의 신뢰성을 위해 샤드들을 복제
- replica는 서로 다른 노드에 존재할 것을 권장

Shard & Replicas(복제본)

분산 검색엔진에서 **shard**란 일종의 파티션과 같은 의미이며, 데이터를 저장할 때 나누어진 하나의 조각에 대한 단위



리플리카는 운영중에 그 수를 변경할수 있으나 샤드는 수를 조정 할수 없음

용어 : Replicas

- '레플리카 샤드' 또는 '레플리카'로 불리는 것들은 여러분의 인덱스의 샤드에 대한 한개 이상의 복사본
- 노드가 깨졌을 때를 대비하여 높은 가용성을 제공
- 레플리카는 검색이 모든 레플리카에서 병렬적으로 실행될 수 있게 해주기 때문에 검색의 볼륨 스케일업을 해줄 수 있음

샤딩(Sharding)을 적용하기에 앞서

- 샤딩(Sharding)을 적용 한다는 것은?
 - 프로그래밍, 운영적인 복잡도는 더 높아지는 단점이 있음
- 가능하면 Sharding을 피하거나 지연시킬 수 있는 방법을 찾는 것이 우선되어야 함
 - Scale-in
 - Hardware Spec이 더 좋은 컴퓨터를 사용
 - Read 부하가 크다면?
 - Cache나 Database의 Replication을 적용하는 것도 하나의 방법
 - Table의 일부 컬럼만 자주 사용한다면?
 - Vertically Partition도 하나의 방법
 - Data를 Hot, Warm, Cold Data로 분리하는 것

용어: 버킷(Bucket)

- 데이터를 일정한 기준으로 나눠서 Grouping 작업
- 히스토그램에서 사용되는 용어
 - "히스토그램에서 동일한 크기의 구간을 몇 개로 잡을 것이냐" 를 결정하는 요소
 - 각 버킷은 동일한 값 범위를 표현
- 예시
 - x값의 범위가 0~10 인 히스토그램 이라 가정
 - 히스토그램의 버킷이 1인 경우
 - 히스토그램의 버킷이 10이라면 한 버킷의 구간은 1의 크기 보유
 - 한 버킷당 0-1, 1-2, 2-3, ... , 9-10 이 된다는 것 의미
 - 버킷의 구간은 0-10, 버킷이 2면
 - 버킷의 구간이 0-5, 6-10

| Elasticsearch Architecture

ElasticSearch Architecture

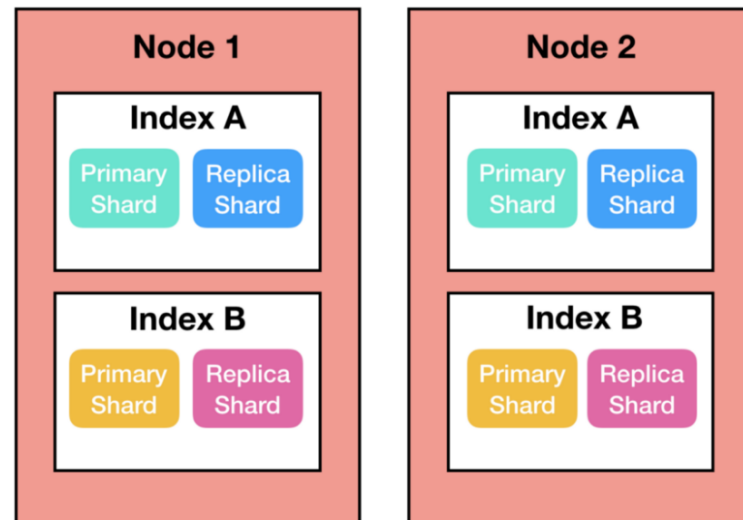
- ElasticSearch 관점
 - Document : 단일 데이터 단위
 - Index : Document를 모아 놓은 집합
 - Indices : 데이터 저장 단위인 index를 의미하기도 함
 - 색인 : Elasticsearch에 저장하는 행위

ElasticSearch Architecture

- ElasticSearch Cluster

- 가장 큰 시스템 단위
- 하나 이상의 노드로 구성
- 여러대의 서버가 하나의 클러스터 구성 가능
- 한 서버에 여러 개의 클러스터 존재 가능

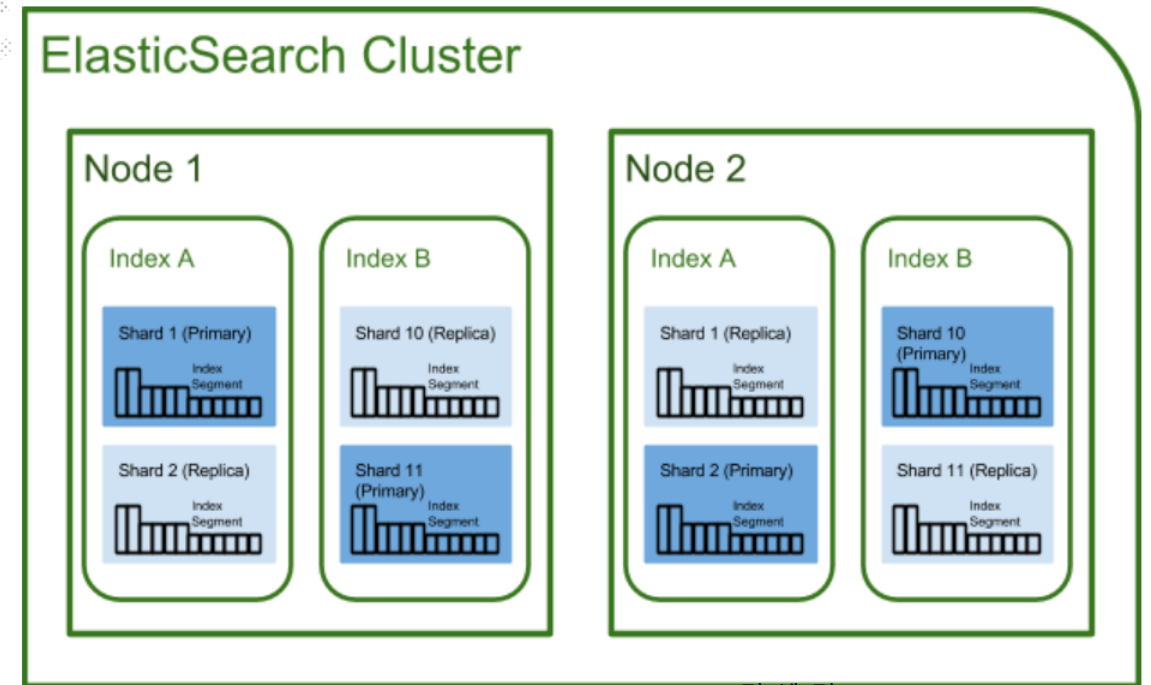
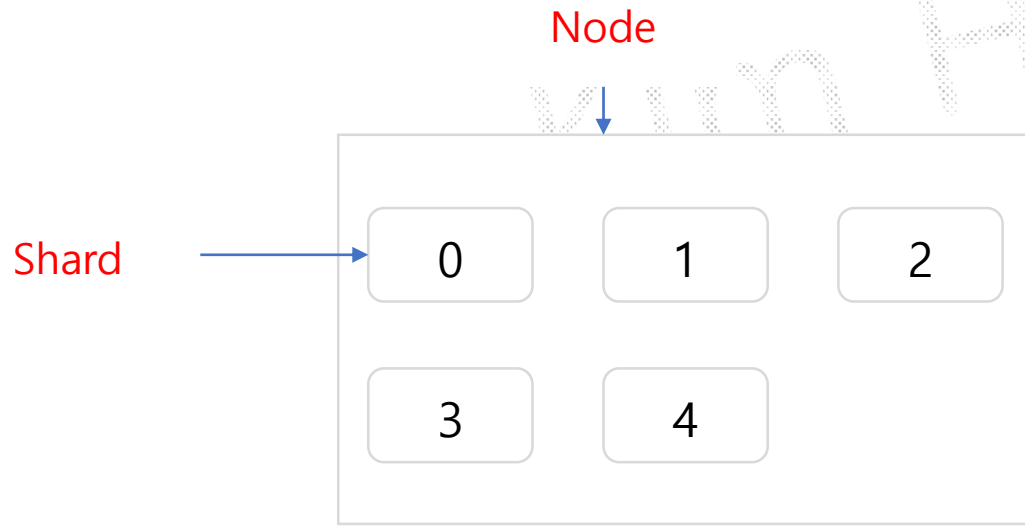
Elasticsearch Cluster



- 서로 다른 클러스터는 데이터의 접근, 교환 불가능한 독립적인 시스템으로 유지
- 클러스터에 있는 노드는 실시간으로 추가, 제거 가능 따라서 가용성이나 확장성 측면에서 매우 유연

ElasticSearch Architecture

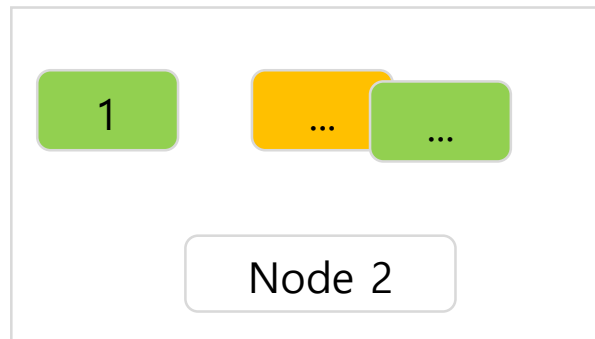
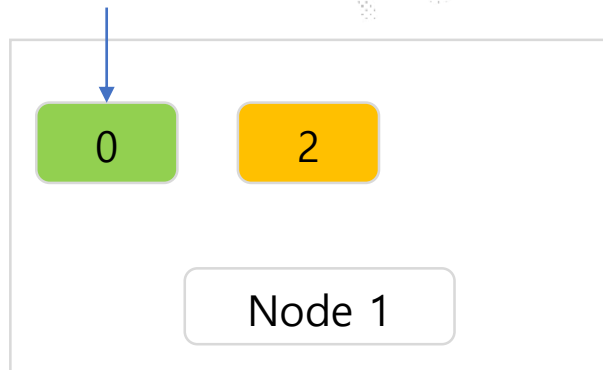
- 노드(node)
 - Elasticsearch를 구성하는 하나의 단위 프로세스
 - index는 기본적으로 shard 라는 단위로 분리되고 각 노드에 분산 저장됨
 - 역할에 따라 Master node, Data node등으로 구분



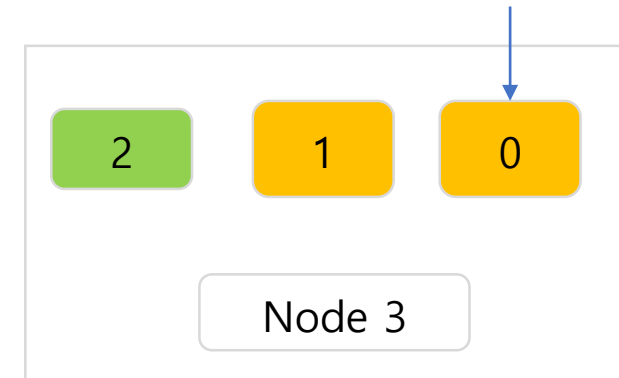
ElasticSearch Architecture

- 프라이머리 샤드(Primary Shard)와 복제본(Replica)
 - cluster에 노드를 추가하게 되면 shard들이 각 node들로 분산되고 default로 1개의 복제본(Replica) 생성
 - Primary Shard : 처음 생성된 shard
 - Replica : 복제본

Primary Shard



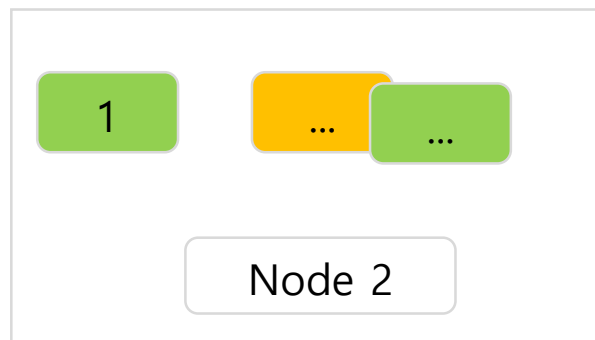
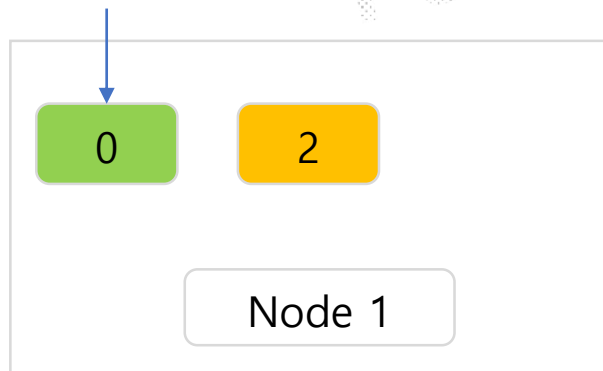
Replica



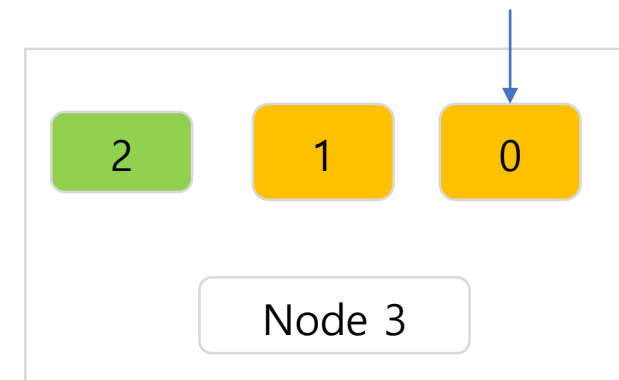
ElasticSearch Architecture

- 같은 Shard와 Replica는 동일한 데이터 보유
- 반드시 서로 다른 Node에 저장되어야 함
 - Node 3이 다운 또는 네트워크 이상으로 소멸되는 경우 Node 3에 있던 0, 1, 2 shard 유실
 - 그러나 node 1, node2에 잔존되어 있기 때문에 데이터는 유실없이 사용 가능

Primary Shard

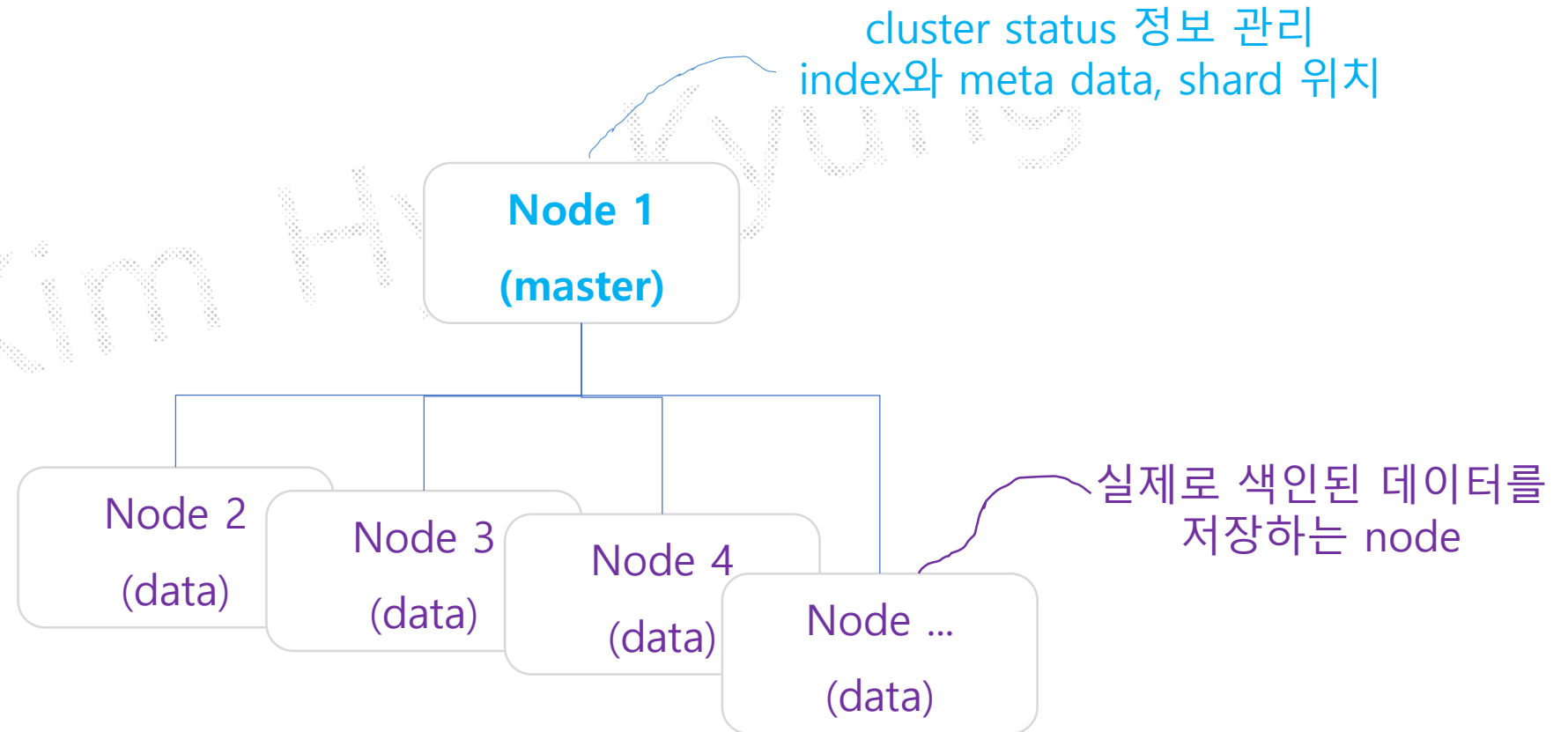


Replica



ElasticSearch Architecture : Node

- ElasticSearch Cluster 관리 기술
 - Shard & Replica 관리 및 데이터 관리 기술
 - 구성
 - Master Node
 - Data Node



ElasticSearch Architecture : Node

Master Node

- 클러스터를 제어 및 관리하는 마스터
- 인덱스 생성, 삭제
- 클러스터 노드들의 추적 및 관리
- 데이터 입력시에 어느 샤드에 할당할 것인지 지정

Coordination Node

- 요청을 단순히 라운드 로빈 방식으로 분산시켜주는 도구
- 사용자의 요청만 받음
- cluster에 관련된 것은 Master node에게 넘기고, 데이터에 관련된 것은 Data Node에 넘기

Data Node

- 문서가 실제로 저장되는 노드
- 데이터와 관련된 CRUD
- 데이터가 실제로 분산 저장되는 물리적 공간인 샤드가 배치되는 노드
- 색인 작업은 CPU, 메모리, 스토리지 같은 컴퓨팅 리소스를 많이 소모 따라서 리소스 모니터링이 필요
- 마스터 노드와 분리해서 구성 권장

Ingest Node

- 색인에 앞서 데이터를 전처리 하기 위한 node
- 인덱스 생성 전 문서의 형식 변경을 다양하게 할 수 있음

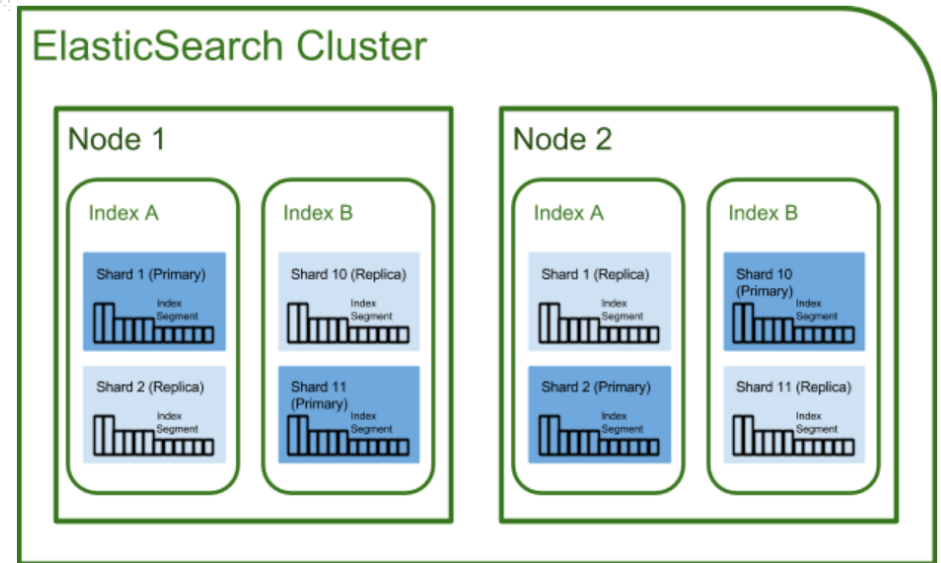
ElasticSearch Architecture : Node

- 예시 :
 - 인덱스별로 shards가 2개
 - replica 1개로 설정

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

ElasticSearch Architecture

- 하나의 ES(ElasticSearch) 클러스터에 총 2개의 물리적인 노드 존재
- ES cluster
 - 인덱스의 문서 조회시 master node를 통해 2개의 node 모두 조회해서 각 데이터를 취합한 후 결과값을 하나로 합쳐서 제공
- Scale out
 - 샤드를 통해 규모가 수평적으로 늘어날 수 있음
- 고가용성
 - Replica를 통해 데이터의 안정성을 보장



ElasticSearch 클러스터 상태

- 클러스터 상태
 - ES를 설치하고 클러스터 상태를 보니, yellow 상태
 - yellow 상태는 모든 데이터의 읽기/쓰기가 가능한 상태이지만 일부 replica shard가 아직 배정되지 않은 상태를 의미
 - 즉, 정상 작동중이지만 replica shard가 없기 때문에 검색 성능에 영향이 있을 수 있음

| Elasticsearch status

ElasticSearch status

- Status란?
 - 현재 클러스터의 상태 표시
 - cat API를 사용하여 손쉽게 클러스터 상태 확인 가능
 - green & yellow
 - 모든 index의 읽기, 쓰기에는 이상이 없는 상태

cat health

health is a terse, one-line representation of the same information from `/_cluster/health`.

```
GET /_cat/health?v
```

[Copy as cURL](#) [View in Console](#)

epoch	timestamp	cluster	status	node.total	node.data	shards	pri	relo
1475871424	16:17:04	elasticsearch	green	1	1	1	1	0

It has one option `ts` to disable the timestamping:

```
GET /_cat/health?v&ts=false
```

[Copy as cURL](#) [View in Console](#)

which looks like:

cluster	status	node.total	node.data	shards	pri	relo	init	unassign	pending_t
elasticsearch	green	1	1	1	1	0	0	0	

<https://www.elastic.co/guide/en/elasticsearch/reference/7.1/cat-health.html>

ElasticSearch status

- status 값 3가지

값	설 명
green	모든 shard가 정상적으로 동작하는 상태 모든 인덱스에 쓰기, 읽기가 정상적으로 동작
yellow	일부 혹은 모든 인덱스의 replicas 샤드가 정상적으로 동작하고 있지 않은 상태, 모든 인덱스에 쓰기/읽기가 정상적으로 동작하지만, 일부 인덱스의 경우 replicas가 없어서 primary 샤드에 문제가 생기면 데이터 유실이 발생할 가능성이 있음
red	일부 혹은 모든 인덱스의 primary와 replicas 샤드가 정상적으로 동작하고 있지 않은 상태, 일부 혹은 모든 인덱스에 쓰기, 읽기가 정상적으로 동작하지 않으며, 데이터의 유실이 발생 할 가능성이 있음

| Elasticsearch 환경 구축

ElasticSearch 실습 환경 구축

- 자바 언어로 개발된 프로그램

- 자바 실행 환경 필요

- 설치 방법

- 다운로드
 - 압축해제
 - 실행

1

Download and unzip Elasticsearch.

!

Elasticsearch can also be installed from our package repositories using apt or yum, or installed on Windows using an MSI installer package. See [Repositories in the Guide](#).

2

Run `bin/elasticsearch` (or `bin\elasticsearch.bat` on Windows)

3

Run `curl http://localhost:9200/` or `Invoke-WebRequest http://localhost:9200` with PowerShell

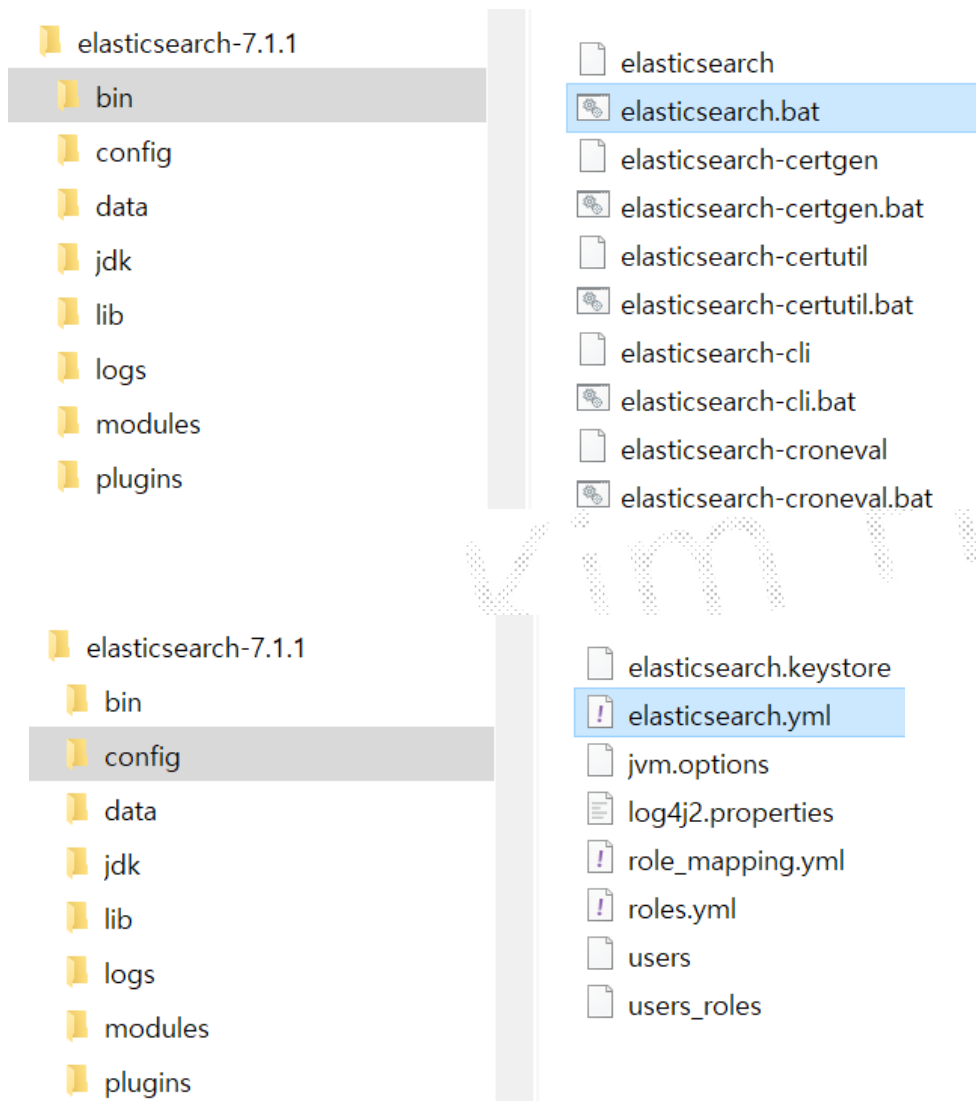
4

Dive into the [getting started guide](#) and [video](#).

ung

<https://www.elastic.co/kr/downloads/elasticsearch>

ElasticSearch 설치 디렉토리



bin : 실행 파일 디렉토리

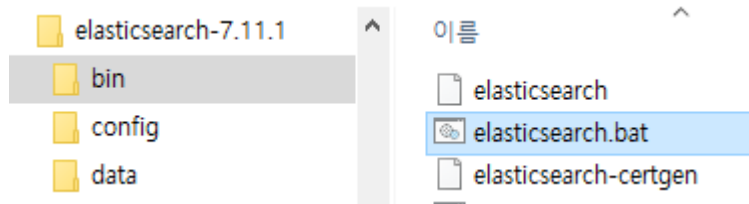
elasticsearch.bat : 실행 파일

elasticsearch.yml : 주 설정 파일

jvm.options : java 설정 파일

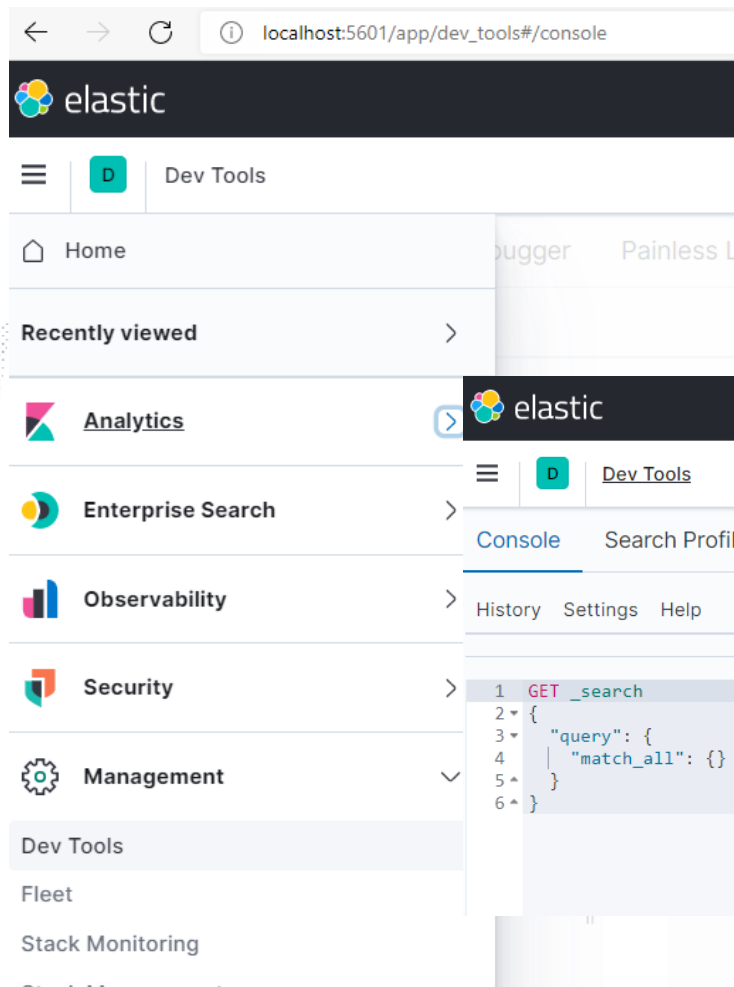
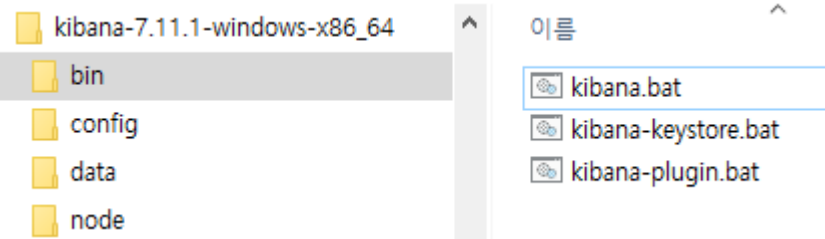
ElasticSearch 실행

1단계 : elasticsearch.bat 실행

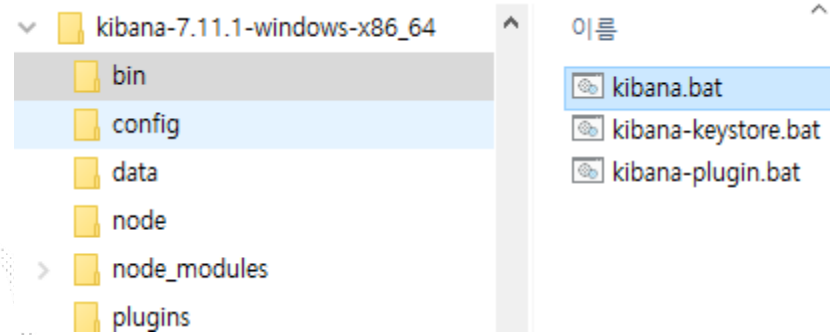


```
{
  "name" : "DESKTOP-R221680",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "DnZacxnRR-4PBsrZHDQrQ",
  "version" : {
    "number" : "7.11.1",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "ff17057114c2199c9c1bbecc727003a907c0db7a",
    "build_date" : "2021-02-15T13:44:09.394032Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

2단계 : kibana.bat



ElasticSearch



| Elasticsearch plug in

- Head

ElasticSearch Head

- <http://mobz.github.io/elasticsearch-head/>
- head plugin 을 이용해서 cluster 상태, index 정보, 간단한 쿼리 수행 등의 기능을 편리하게 사용할 수 있음

The screenshot displays the ElasticSearch Head web interface. At the top, the URL is `http://localhost:9200/` and the cluster health is **yellow (12 of 12 nodes available)**. The main section is titled **Cluster Overview** and shows a table of indices. The indices listed are `.kibana_1`, `.kibana_task_manager`, `bank`, `classes`, and `custom`. Each index entry shows its size and document count, along with **Info** and **Actions** buttons. Below the index list, there is a section for **Unassigned** shards, showing a table with columns for the index name and shard number. The first row shows `DESKTOP-CG56BAI` with shard `0` in a green box, indicating it is assigned. The second row shows `bank` with shard `0` in a green box, indicating it is assigned. The third row shows `classes` with shard `0` in a green box, indicating it is assigned. The fourth row shows `custom` with shard `0` in a green box, indicating it is assigned.

bank의 Actions 버튼에서 삭제 가능
post맨 통해서 push하면 다시 생성됨

ElasticSearch Head

- Browser tab을 이용한 bank index 상세 보기

Elasticsearch **elasticsearch** **cluster health: yellow (15 of 25)**

Browser

All Indices ▾

INDICES

- .kibana_1
- .kibana_task_manager
- bank**
- chapter
- classes
- emails
- kibana_sample_data_ecommerce
- kibana_sample_data_flights
- kibana_sample_data_logs
- logstash-2015.05.18
- logstash-2015.05.19
- logstash-2015.05.20
- shakespeare

Searched 1 of 1 shards. 1000 hits. 0.002 seconds

_index	_type	_id	_score ▲	account_number	balance	firstname	lastname	age	gender	address	employer	email
bank	_doc	1	1	1	39225	Amber	Duke	32	M	880 Holmes Lane	Pyrami	amberduke@pyrami.com
bank	_doc	6	1	6	5686	Hattie	Bond	36	M	671 Bristol Street	Netagy	hattiebond@netagy.com
bank	_doc	13	1	13	32838	Nanette	Bates	28	F	789 Madison Street	Quility	nanettebates@quility.com
bank	_doc	18	1	18	4180	Dale	Adams	33	M	467 Hutchinson Court	Boink	daleadams@boink.com
bank	_doc	20	1	20	16418	Elinor	Ratliff	36	M	282 Kings Place	Scentric	elinorratliff@scentric.com
bank	_doc	25	1	25	40540	Virginia	Ayala	39	F	171 Putnam Avenue	Filodyne	virginiaayala@filodyne.com
bank	_doc	32	1	32	48086	Dillard	Mcpherson	34	F	702 Quentin Street	Quailcom	dillardmcpherson@quailcom.com
bank	_doc	37	1	37	18612	Mcgee	Mooney	39	M	826 Fillmore Place	Reversus	mcgeemooney@reversus.com
bank	_doc	44	1	44	34487	Aurelia	Harding	37	M	502 Baycliff Terrace	Orbalix	aureliaharding@orbalix.com
bank	_doc	49	1	49	29104	Fulton	Holt	23	F	451 Humboldt Street	Anocha	fultonholt@anocha.com
bank	_doc	51	1	51	14097	Burton	Meyers	31	F	334 River Street	Bezal	burtonmeyers@bezal.com
bank	_doc	56	1	56	14992	Josie	Nelson	32	M	857 Tabor Court	Emtrac	josienelson@emtrac.com
bank	_doc	63	1	63	6077	Hughes	Owens	30	F	510 Sedgwick Street	Valpreal	hughesowens@valpreal.com

ElasticSearch Head

- Any Request(+) tab을 이용한
bank index 정보 보기

Elasticsearch http://localhost:9200/ Connect **elasticsearch**

Overview Indices Browser Structured Query [+] Any Request [+]

History

Query

http://localhost:9200/

bank/_doc/1 GET

```
{
  "settings": {
    "index": {
      "number_of_shards": 3,
      "number_of_replicas": 2
    }
  }
}
```

Request Validate JSON ☒ Pretty

```
{
  "_index": "bank",
  "_type": "_doc",
  "_id": "1",
  "_version": 2,
  "_seq_no": 1000,
  "_primary_term": 4,
  "found": true,
  "_source": {
    "account_number": 1,
    "balance": 39225,
    "firstname": "Amber",
    "lastname": "Duke",
    "age": 32,
    "gender": "M",
    "address": "880 Holmes Lane",
    "employer": "Pyrami",
    "email": "amberduke@pyrami.com",
    "city": "Brogan",
    "state": "IL"
  }
}
```

Elasticsearch http://localhost:9200/ Connect **elasticsearch** cluster health: yellow (15 of 25)

Overview Indices Browser Structured Query [+] Any Request [+]

History

Query

http://localhost:9200/

bank GET

```
{
  "settings": {
    "index": {
      "number_of_shards": 3,
      "number_of_replicas": 2
    }
  }
}
```

Request Validate JSON ☒ Pretty

Result Transformer ?

Repeat Request

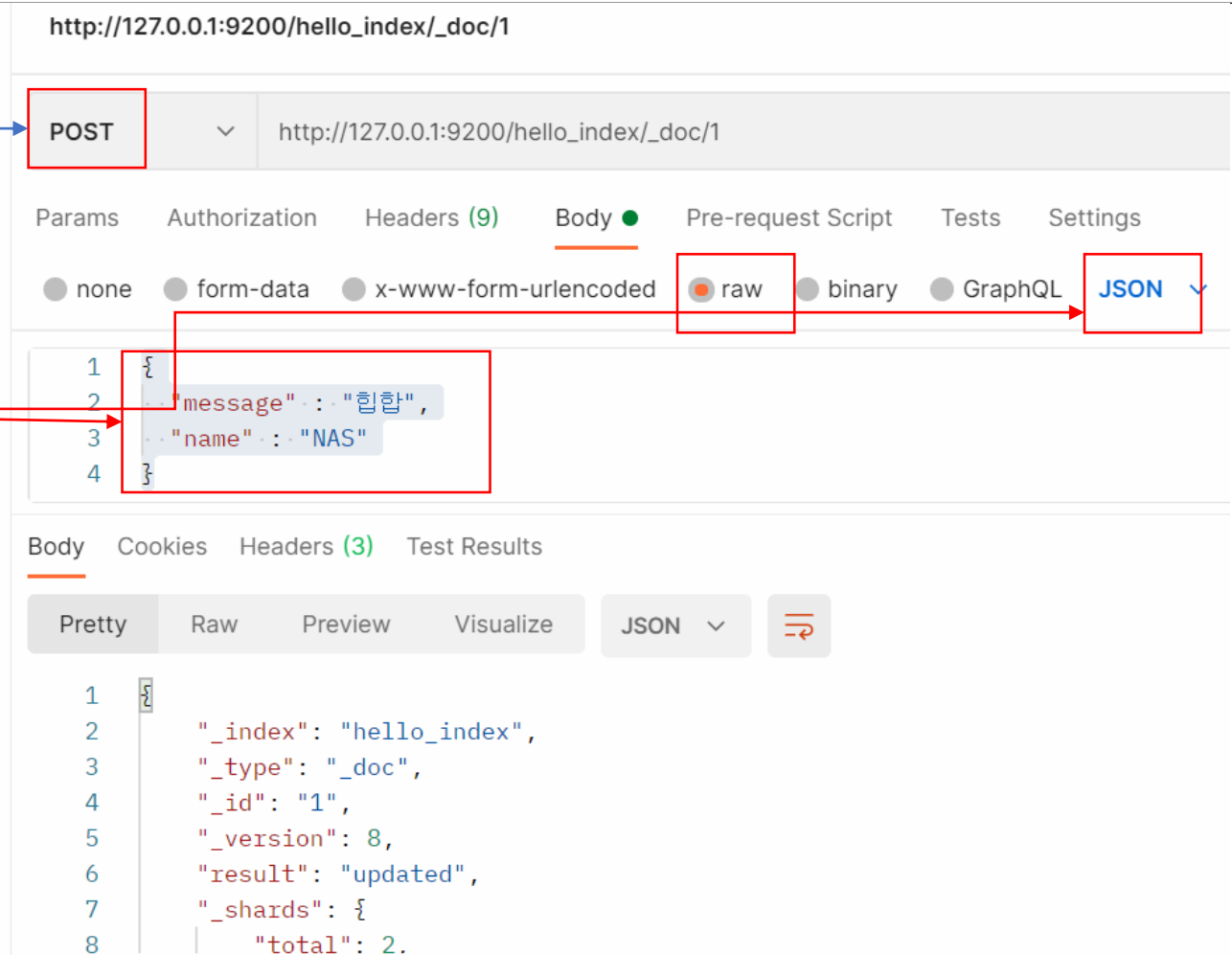
Display Options ?

```
{
  "bank": {
    "aliases": { },
    "mappings": {
      "properties": {
        "account_number": {
          "type": "long"
        },
        "address": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "age": {
          "type": "long"
        },
        "balance": {
          "type": "long"
        },
        "city": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "email": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "employer": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
      }
    }
  }
}
```


Postman 사용 방법

```
8
9 # table 생성 및 데이터 저장
10 POST hello_index/_doc/1
11 {
12   "message" : "힙힙",
13   "name" : "NAS"
14 }
15
16 # select와 동일한 검색
17 GET hello_index/_doc/1
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
1 {
2   "_index" : "hello_index",
3   "_type" : "_doc",
4   "_id" : "1",
5   "_version" : 7,
6   "result" : "updated",
7   "_shards" : {
8     "total" : 2,
9     "successful" : 1,
10    "failed" : 0
11  },
12  "_seq_no" : 6,
13  "_primary_term" : 1
14 }
```



```
# table 생성 및 데이터 저장
POST hello_index/_doc/1
{
  "message" : "힙합",
  "name" : "NAS"
}
```

```
# select와 동일한 검색
GET hello_index/_doc/1
```

```
# 데이터 수정
PUT hello_index/_doc/1
{
  "message" : "유재석"
}
GET hello_index/_doc/1
```

```
# pk 1 즉 id가 1인 데이터 삭제
DELETE hello_index/_doc/1
```

The screenshot shows a REST client interface with a top navigation bar containing 'Home', 'Workspaces', 'Reports', and 'Explore'. A search bar is on the right. The main area displays a POST request to 'http://127.0.0.1:9200/hello_index/_doc/1'. The request body is a JSON object: `{ "message": "힙합", "name": "NAS" }`. Below the request, the response is shown in the 'Body' tab, displaying a JSON object: `{ "_index": "hello_index", "_type": "_doc", "_id": "1", "_version": 1, "result": "created", "_shards": { "total": 2, "successful": 1, ... } }`. The interface includes tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is active, and the response is formatted as JSON.

```
# table 생성 및 데이터 저장
POST hello_index/_doc/1
{
  "message" : "힙힙",
  "name" : "NAS"
}
```

```
# select와 동일한 검색
GET hello_index/_doc/1
```

```
# 데이터 수정
PUT hello_index/_doc/1
{
  "message" : "유재석"
}
GET hello_index/_doc/1
```

```
# pk 1 즉 id가 1인 데이터 삭제
DELETE hello_index/_doc/1
```

GET http://127.0.0.1:9200/hello_index/_doc/1

GET http://127.0.0.1:9200/hello_index/_doc/1

Params Authorization Headers (7) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "_index": "hello_index",
3   "_type": "_doc",
4   "_id": "1",
5   "_version": 1,
6   "_seq_no": 10,
7   "_primary_term": 1,
8   "found": true,
9   "_source": {
10     "message": "힙힙",
11     "name": "NAS"
```

table 생성 및 데이터 저장

POST hello_index/_doc/1

```
{  
  "message" : "힙힙",  
  "name" : "NAS"  
}
```

select와 동일한 검색

GET hello_index/_doc/1

데이터 수정

PUT hello_index/_doc/1

```
{  
  "message" : "유재석"  
}
```

GET hello_index/_doc/1

pk 1 즉 id가 1인 데이터 삭제

DELETE hello_index/_doc/1

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:9200/hello_index/_doc/1`. The request body is a JSON object: `{ "message" : "유재석" }`. The response is also in JSON format, showing document details: `{ "_index": "hello_index", "_type": "_doc", "_id": "1", "_version": 2, "result": "updated", "_shards": { "total": 2, "successful": 1, "failed": 0 } }`. The interface includes tabs for Params, Authorization, Headers (9), Body, Pre-request Script, Tests, and Settings. The Body tab is active, showing the request and response bodies in a code editor.

```
# table 생성 및 데이터 저장
POST hello_index/_doc/1
{
  "message" : "힙힙",
  "name" : "NAS"
}
```

```
# select와 동일한 검색
GET hello_index/_doc/1
```

```
# 데이터 수정
PUT hello_index/_doc/1
{
  "message" : "유재석"
}
GET hello_index/_doc/1
```

```
# pk 1 즉 id가 1인 데이터 삭제
DELETE hello_index/_doc/1
```

Home Workspaces Reports Explore

DEL http://127.0.0.1:92... + ...

http://127.0.0.1:9200/hello_index/_doc/1

DELETE http://127.0.0.1:9200/hello_index/_doc/1

Params Authorization Headers (7) Body Pre-request Script Tests Setting

none form-data x-www-form-urlencoded raw binary GraphQL

This requ

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```

1  {
2    "_index": "hello_index",
3    "_type": "_doc",
4    "_id": "1",
5    "_version": 3,
6    "result": "deleted",
7    "_shards": {
8      "total": 2,
9      "successful": 1,
10     "failed": 0
11   },
12   "_seq_no": 12,

```

참고 문헌 및 Site

- https://db-engines.com/en/ranking_trend/search+engine

Kim Hye Kyung