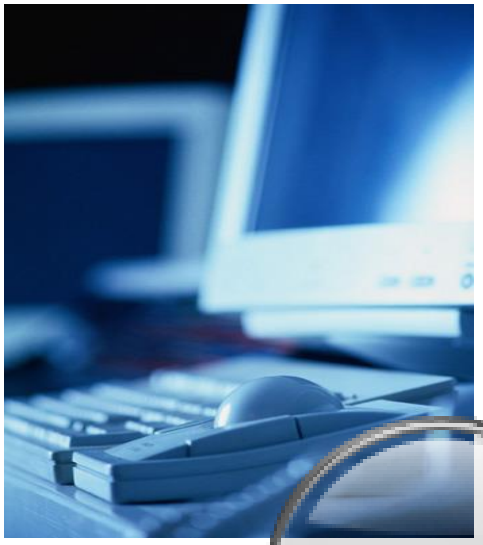
데이터 관리를 위한 엘라스틱서치 기초

- 엘라스틱서치 소개와 설치
- 데이터 입력과 조회
- 배치 프로세스
- 검색 API
- KIBANA 소개와 설치
- KIBANA 데이터 준비
- KIBANA 시각화
- 파일 비트를 활용한 아파치 서버 로그 수집



엘라스틱서치 소개와 설치

엘라스틱서치!

- 확장성이 뛰어난 오픈 소스 전체 텍스트 검색 및 분석 엔진
- 대량의 데이터를 신속하고 거의 실시간으로 저장, 검색 및 분석
- 일반적으로 복잡한 검색 기능과 요구 사항이 있는 응용 프로그램을 구동하는 기본 엔진 / 기술

사용 사례

- 고객이 판매하는 제품을 검색 할 수 있는 **온라인 웹 스토어를 운영**
  - 이 경우 Elasticsearch를 사용하여 **전체 제품 카탈로그 및 인벤토리를 저장하고 검색 및 자동 완성** 제안을 제공

- **로그 또는 트랜잭션 데이터**를 수집, 분석 및 조사하여 **추세, 통계, 요약 또는 예외를 탐지**
  - 이 경우 Logstash (Elasticsearch / Logstash / Kibana 스택의 일부)를 사용하여 데이터를 수집, 집계 및 구문 분석한 다음 Logstash에 이 데이터를 Elasticsearch에 제공
  - 데이터가 Elasticsearch에 저장되면 검색 및 집계를 실행하여 관심 있는 정보를 검색

elastic

사용 사례

- 가격에 정통한 고객이 "특정 전자 장치를 구입하는 데 관심이 있으며 다음 달의 모든 공급 업체 가젯 가격이 $ X 이하로 떨어지면 알림을 받고 싶습니다"와 같은 규칙을 지정할 수 있는 **가격 알림 플랫폼을 운영**
  - 이 경우 공급 업체 가격을 긁어내어 Elasticsearch로 밀어 넣은 뒤
  - 역방향 검색 (Percolator) 기능을 사용하여 가격 변동을 고객 쿼리와 비교하고 일치 항목이 발견
  - 마지막으로 고객에게 알리미를 전송

- 분석 / 비즈니스 인텔리전스 요구 사항이 있으며 **많은 데이터에 대해 신속하게 조사, 분석, 시각화 및 임시 질문**을 하고 싶습니다 (수억 또는 수십억 건의 레코드를 생각하십시오).
  - Elasticsearch 를 활용해 데이터를 저장 한 다음 Kibana (Elasticsearch / Logstash / Kibana 스택의 일부)를 사용하여 중요한 데이터를 시각화 할 수 있는 사용자 정의 대시 보드를 작성
  - 또한 Elasticsearch 집계 기능을 사용하여 데이터에 대해 복잡한 비즈니스 인텔리전스 쿼리를 수행

Elasticsearch의 핵심 개념

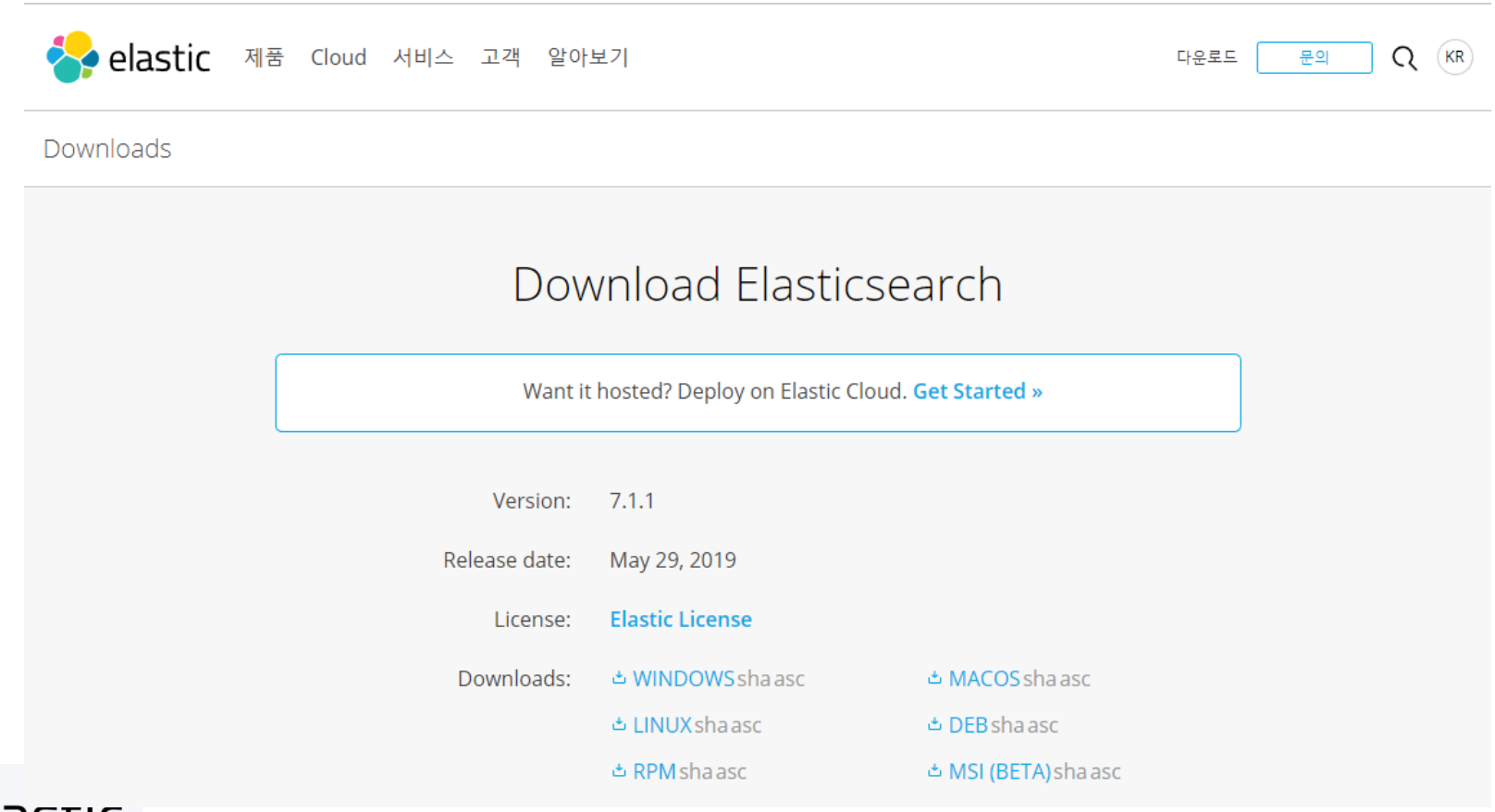
Near Realtime (NRT)	Elasticsearch는 거의 실시간 검색 플랫폼 문서를 색인할 때부터 검색 가능할 때까지 약간의 대기 시간 (일반적으로 1 초)이 매우 짧음
클러스터 (Cluster)	전체 데이터를 함께 보유하고 모든 노드에서 연합 인덱싱 및 검색 기능을 제공하는 하나 이상의 노드 (서버) 모음 클러스터는 기본적으로 "elasticsearch"라는 고유 한 이름으로 식별 이 이름은 노드가 이름으로 클러스터에 참여하도록 설정된 경우 노드가 클러스터의 일부일 수 있기 때문에 중요
노드 (Node)	노드는 클러스터의 일부이며 데이터를 저장하고 클러스터의 인덱싱 및 검색 기능에 참여하는 단일 서버 단일 클러스터에서 원하는 만큼의 노드를 소유 가능 또한 현재 네트워크에서 실행중인 다른 Elasticsearch 노드가 없는 경우 단일 노드를 시작하면 기본적으로 elasticsearch라는 새로운 단일 노드 클러스터가 형성
색인 (Index)	색인은 다소 유사한 특성을 갖는 문서의 콜렉션 예를 들어, 고객 데이터에 대한 색인, 제품 카탈로그에 대한 또 다른 색인 및 주문 데이터에 대한 또 다른 색인을 가질 수 있음 색인은 이름(모두 소문자 여야 함)로 식별되며 이 이름은 색인 된 문서를 색인 작성, 검색, 갱신 및 삭제할 때 색인을 참조하는 데 사용

Elasticsearch의 핵심 개념

Type	사용자가 하나의 유형, 블로그 게시물을 다른 유형과 같이 여러 Type의 문서를 동일한 색인에 저장할 수 있도록 색인의 논리적 범주 / 파티션으로 사용되는 유형 더 이상 인덱스에 여러 유형을 작성할 수 없으며 이후 버전에서는 Type의 전체 개념이 제거됩니다.
Documents	문서는 색인을 생성 할 수있는 기본 정보 단위 예를 들어, 단일 고객에 대한 문서, 단일 제품에 대한 다른 문서 및 단일 주문에 대한 문서를 보유 JSON (JavaScript Object Notation)으로 표현
RESTFul API	URI를 사용한 동작이 가능 HTTP 프로토콜로 JSON 문서의 입출력과 다양한 제어 JSON 문서의 입출력과 다양한 제어

Elasticsearch 다운로드

- https://www.elastic.co/kr/downloads/elasticsearch



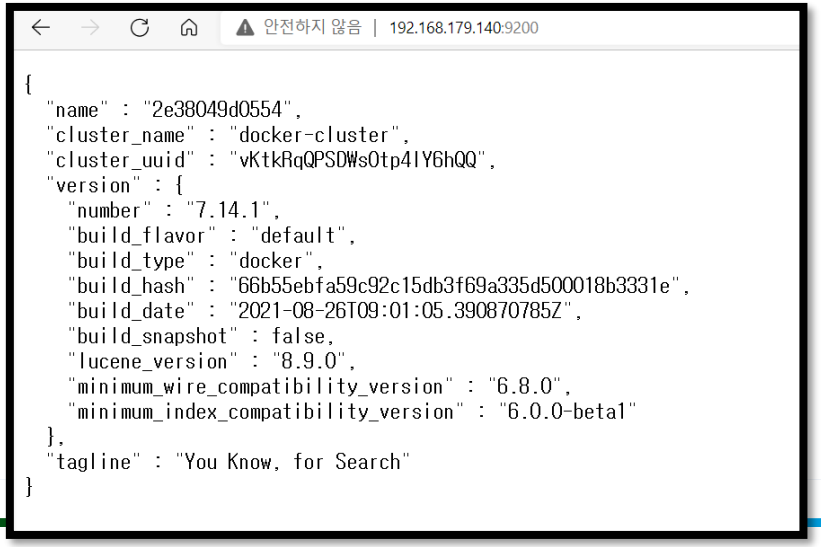
elastic

도커를 활용한 Elasticsearch 설치하기

- https://hub.docker.com/_/elasticsearch
- 우분투에 docker.io를 설치하고 엘라스틱서치와 키바나를 설치

```
sudo apt update && sudo apt install -y docker.io
docker network create elastic
docker run -d --name es01-test --net elastic -p 9200:9200 -p 9300:9300 -e
"discovery.type=single-node" elasticsearch:7.14.1
docker run -d --name kib01-test --net elastic -p 5601:5601 -e
"ELASTICSEARCH_HOSTS=http://es01-test:9200" kibana:7.14.1
```

- 해당 이미지의 9200번 포트로 HTTP 접속



Elasticsearch 도커 설치

- 도커 현재 상태 및 로그 확인 방법

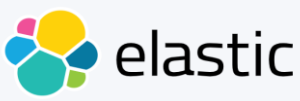
```
docker ps -a
docker logs es01-test
docker logs kib01-test
```

- 도커 컨테이너 종료 방법

```
docker stop es01-test # 시작할 때는 start 명령을 사용
docker stop kib01-test
```

- 도커 컨테이너 삭제 방법

```
docker network rm elastic
# 컨테이너 삭제
docker rm es01-test
docker rm kib01-test
# 이미지 삭제
docker rmi docker.elastic.co/elasticsearch/elasticsearch:7.15.0
docker rmi docker.elastic.co/kibana/kibana:7.15.0
```



데이터 입력과 조회

REST API

- 웹의 창시자(HTTP) 중의 한 사람인 Roy Fielding의 2000년 논문에 의해서 소개
  "웹의 장점을 최대한 활용할 수 있는 네트워크 기반의 아키텍쳐"
- 엘라스틱서치 노드와 통신하는 방법
- 클러스터와 상호 작용하는 데 사용할 수 있는 매우 포괄적이고 강력한 REST API를 제공
- API로 수행 할 수 있는 몇 가지 작업
  - 클러스터, 노드 및 색인 상태, 상태 및 통계 확인
  - 클러스터, 노드 및 색인 데이터 및 메타 데이터 관리
  - 데이터 입력과 검색 (Create, Read, Update, Delete) 및 인덱스에 대한 검색 작업 수행
  - 페이징, 정렬, 필터링, 스크립팅, 집계 및 기타 여러 고급 검색 작업 실행

REST API 구성요소 3가지

- 리소스, 메서드, 메시지

elastic

클러스터 상태(Health)

- 클러스터가 어떻게 진행되고 있는지 기본적인 확인
- 우리는 curl을 사용하여 이를 수행
- HTTP/REST 호출을 수행 할 수 있는 모든 도구를 사용 가능
- 클러스터 상태를 확인하기 위해 _cat API를 사용
- 녹색 - 모든 것이 좋음(클러스터가 완전히 작동 함).
- 노란색 - 모든 데이터를 사용할 수 있지만 일부 복제본은 아직 할당되지 않음(클러스터는 완전히 작동

```
GET /_cat/nodes?v
```

- 빨간색 - 어떤 이유로든 일부 데이터를 사용할 수 없음(클러스터가 부분적으로 작동 함).

데이터 입력과 조회

데이터베이스가 가진 데이터 확인하기

- 갖고 있는 모든 인덱스 항목 조회
- index는 일반 RDB에서의 데이터베이스의 역할

```
GET /_cat/indices?v
```

elastic

엘라스틱 데이터베이스의 인덱싱 방식

문서	문서 내용
Doc 1	blue sky green land red sun
Doc 2	blue ocean green land
Doc 3	red flower blue sky

일반적인 관계형 DB
테이블에서 텍스트 저장

검색어	검색어가 가리키는 문서
blue	Doc1, Doc2, Doc3
sky	Doc1, Doc3
green	Doc1, Doc2
land	Doc1, Doc2
red	Doc1, Doc3
ocean	Doc2
flower	Doc3
sun	Doc1

색인 공간에 저장된
역파일 색인 데이터 구조

elastic

데이터 입력과 조회

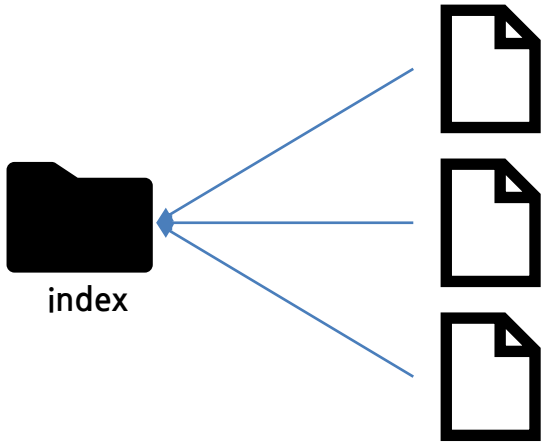
- HTTP 메서드와 데이터 입력과 검색, SQL을 비교

	HTTP 메서드	데이터 입력과 검색	SQL
	GET	Read	Select
	PUT	Update	Update
	POST	Create	Insert
	DELETE	Delete	Delete

엘라스틱서치 데이터 처리

- 엘라스틱서치의 데이터 구조
  - 인덱스(Index), 도큐먼트(Document)의 단위를 가짐
  - 도큐먼트는 엘라스틱서치의 데이터가 저장되는 최소 단위
  - 여러 개의 도큐먼트는 하나의 인덱스로 구성

관계형 DB	엘라스틱서치
데이터베이스(Database)	인덱스(Index)
테이블(Table)	타입(Type, 사라질 예정)
열(Row)	도큐먼트(Document)
행(Column)	필드(Field)
스키마	매핑(Mapping)



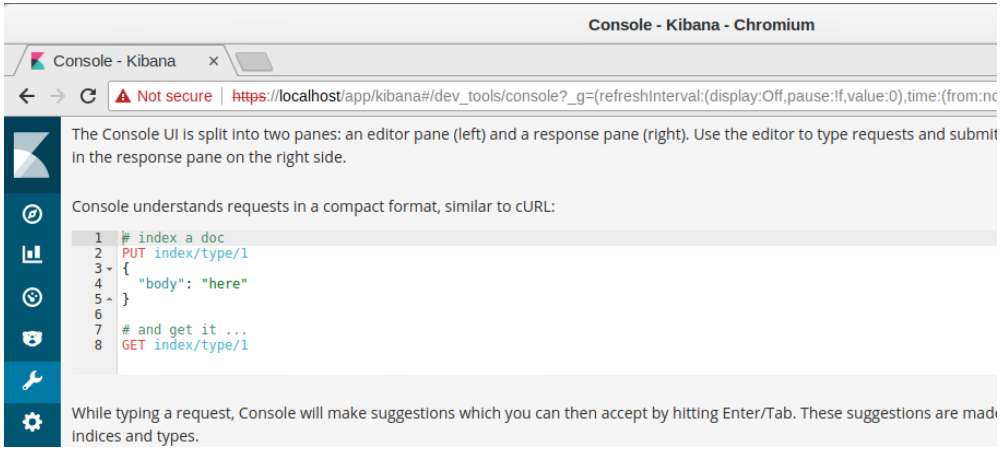
elastic

데이터 입력과 조회

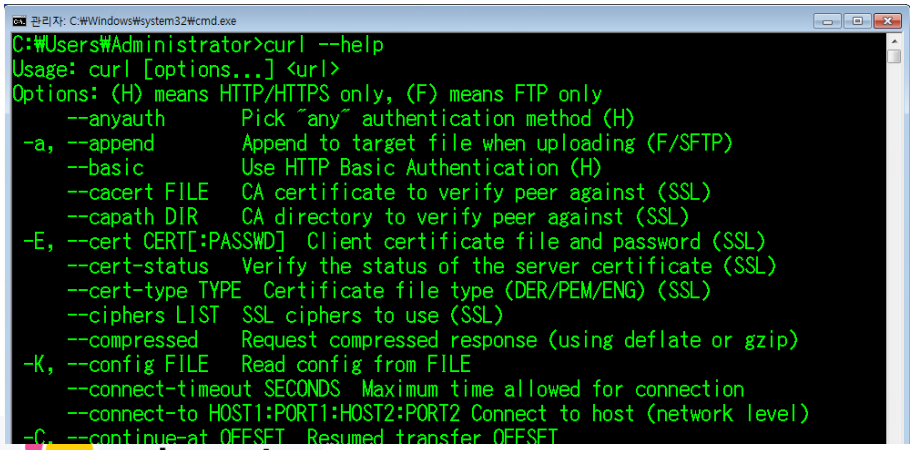
엘라스틱서치의 질의 방법

- 커맨드라인의 curl 명령어 사용
- postman 응용프로그램 사용
- Kiabana에서 devtool 사용

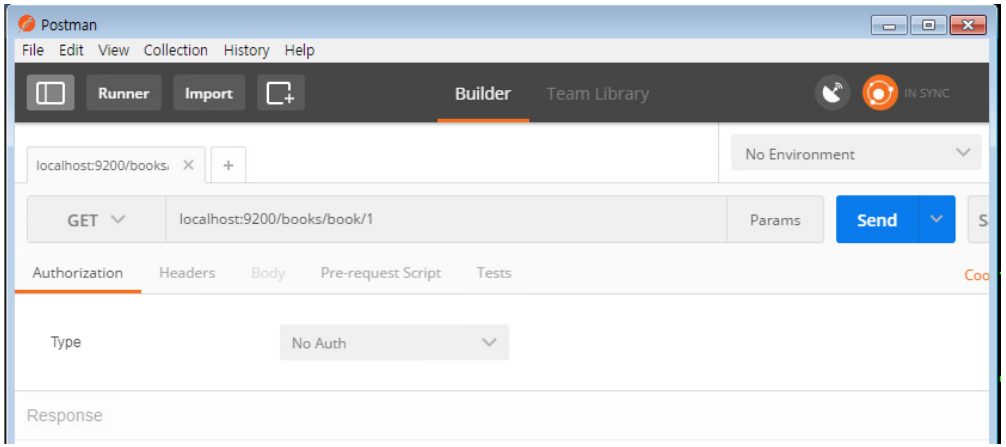
Kibana의 devtool



curl 도움말



Postman 프로그램



elastic

데이터 입력/조회/삭제/업데이트 요약

데이터 처리	메서드	구문
입력	PUT	http://localhost:9200/index1/type1/1 -d '{"num":1, "name":"Ilsun Choi"}'
조회	GET	http://localhost:9200/index1/type1/1
삭제	DELETE	http://localhost:9200/index1/type1/1
업데이트	POST	http://localhost:9200/index1/type1/1/_update -d '{doc: {"age":99} }'

※ 6.x 이상에서는 POST와 PUT을 혼용

elastic

인덱스 만들기

- Customer라는 인덱스를 만들어보자.

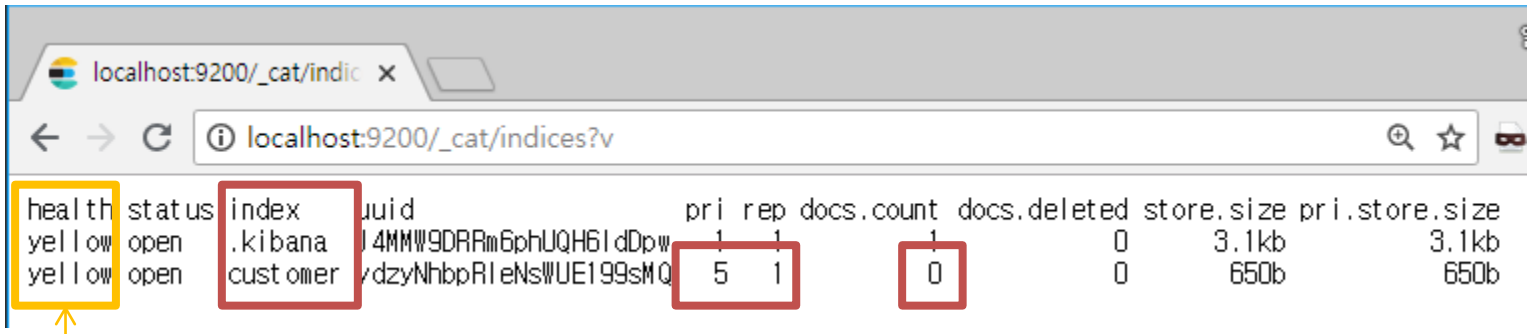
```
PUT /customer?pretty
GET /_cat/indices?v

<curl 명령어>
curl -X PUT "localhost:9200/customer?pretty"
curl -X GET "localhost:9200/_cat/indices?v"
```

- 첫 번째 명령은 PUT 동사를 사용하여 "customer"라는 색인을 작성
- JSON 응답 (있을 경우)을 예쁜 것으로 인쇄하도록 호출의 끝 부분에 간단히 추가

인덱스 만들기

- Customer라는 인덱스를 만들어보자.



도큐먼트 입력과 조회

- costumer 인덱스에 데이터 입력
  - ID가 1 인 간단한 고객 문서를 고객 색인으로 색인화

Console | Search Profiler | Grok Debugger | Painless Lab BETA

History | Settings | Help

```
1  POST customer/_doc/1
2  {
3    "name": "John Doe"
4  }
```

```
1  #! Elasticsearch built-in security features are
     /elasticsearch/reference/7.14/security-minima
2  {
3    "_index" : "customer",
4    "_type" : "_doc",
5    "_id" : "1",
6    "_version" : 2,
7    "result" : "updated",
8    "_shards" : {
9      "total" : 2,
10     "successful" : 1,
11     "failed" : 0
12   },
13   "_seq_no" : 1,
14   "_primary_term" : 1
15 }
16
```

elastic

도큐먼트 입력과 조회

- 입력한 데이터 조회

History Settings Help

```
1  GET customer/_doc/1
```

```
1  #! Elasticsearch built-in security features
   /elasticsearch/reference/7.14/security-m
2  {
3    "_index" : "customer",
4    "_type" : "_doc",
5    "_id" : "1",
6    "_version" : 2,
7    "_seq_no" : 1,
8    "_primary_term" : 1,
9    "found" : true,
10   "_source" : {
11     "name" : "John Doe"
12   }
13 }
14
```

elastic

인덱스 삭제

- 고객 색인에 뭔가 넣기
  - 방금 작성한 색인을 삭제 한 다음 모든 색인을 다시 나열

```
DELETE /customer?pretty
GET /_cat/indices?v
```

```
1 {
2     "acknowledged": true
3 }
```

응답

  - 응답은 인덱스가 성공적으로 삭제되었음을 의미
  - 다음 단계로 넘어 가기 전에 지금까지 배웠던 몇 가지 API 명령을 확인

```
PUT /customer
PUT /customer/_doc/1
{
        "name": "John Doe"
}
GET /customer/_doc/1
DELETE /customer
```

elastic

데이터 입력과 조회

여러 개의 데이터 입력

History Settings Help

```
1  POST books/_doc/1
2  {
3    "title": "Elasticsearch Guide",
4    "author": "Choi",
5    "date" : "2021-10-30",
6    "pages" : 500
7  }
```

```
1  #! Elasticsearch built-in security features are
     /guide/en/elasticsearch/reference/7.14/securit
2  {
3    "_index" : "books",
4    "_type" : "_doc",
5    "_id" : "1",
6    "_version" : 1,
7    "result" : "created",
8    "_shards" : {
9      "total" : 2,
10     "successful" : 1,
11     "failed" : 0
12   },
13   "_seq_no" : 0,
14   "_primary_term" : 1
15 }
16
```

elastic

데이터 입력과 조회

데이터 조회

History Settings Help

```
1  GET books/_doc/1
```

```
1  #! Elasticsearch built-in security features are not ena
     /guide/en/elasticsearch/reference/7.14/security-minir
2  {
3    "_index" : "books",
4    "_type" : "_doc",
5    "_id" : "1",
6    "_version" : 1,
7    "_seq_no" : 0,
8    "_primary_term" : 1,
9    "found" : true,
10   "_source" : {
11     "title" : "Elasticsearch Guide",
12     "author" : "Choi",
13     "date" : "2021-10-30",
14     "pages" : 500
15   }
16 }
17
```

elastic

도큐먼트 삭제

- 문서를 삭제하는 것은 매우 간단
- ID가 2 인 이전 고객을 삭제하는 방법

```
DELETE /customer/_doc/2?pretty
```

- 데이터 삭제 후 데이터 조회 시 found가 false임을 확인

- 데이터 삭제 시 특징
  - 메타데이터가 그대로 유지
  - 도큐먼트 삭제 후 다시 데이터를 입력하면 _version 값이 이어서 진행
  - 버전까지 초기화하려면 인덱스를 삭제해야 함

데이터 수정

- Elasticsearch는 거의 실시간으로 데이터 조작 및 검색 기능을 제공
- 기본적으로 데이터를 색인 / 업데이트 / 삭제할 때부터 검색 결과에 표시 될 때까지 1 초의 지연 (새로 고침 간격)을 기대
- ID를 고쳐 쓰면 모든 내용이 교체됨
- ID를 쓰지 않거나 다른 ID를 사용할 경우 새롭게 저장
- 6.x부터는 POST와 PUT을 혼용

```
PUT /customer/_doc/2?pretty
{
        "name": "Jane Doe"
}
```

```
POST /customer/_doc/?pretty
{
        "name": "Jane Doe"
}
```

도큐먼트 아이디를 명시하지 않으면 랜덤하게 생성

데이터 업데이트

- 입력된 도큐먼트를 수정하는 _update API 제공
- _update API의 두 개의 매개 변수인 doc와 source를 이용해서 데이터를 제어
  - doc : 도큐먼트에 새로운 필드를 추가하거나 기존 필드 값을 변경하는 데 사용
  - script : 프로그래밍 기법을 사용. 입력된 내용에 따라 필드의 값을 변경하는 등의 처리

데이터 입력과 조회

데이터 업데이트 예제

필드 수정 or 추가

필드 수정 or 추가

script 필드 수정

```
POST customer/_doc/1
{
  "name": "John Doe"
}

POST customer/_update/1/
{
  "doc" : {
    "category" : "IT",
    "pages" : 50
  }
}

POST customer/_update/1/
{
  "doc" : {
    "author" : "CHOI"
  }
}

POST customer/_update/1/
{
  "script" : "ctx._source.pages+=50"
}
```

결과

```
{
  "_index" : "customer",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 12,
  "_seq_no" : 18,
  "_primary_term" : 2,
  "found" : true,
  "_source" : {
    "name" : "John Doe",
    "pages" : 100,
    "category" : "IT",
    "author" : "CHOI"
  }
}
```

elastic

데이터 업데이트 예제

- script의 ctx.op 명령을 사용하여 필드 조건에 따라 도큐먼트 삭제

```
POST customer/_update/1/
{
  "script" :{
    "source" : "if(ctx._source.pages <= params
      .page_cnt){ctx.op='delete'} else{ctx.op
      ='none'}",
    "params": {
      "page_cnt" : 150
    }
  }
}
```

```
{
  "_index" : "customer",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 13,
  "result" : "deleted",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 19,
  "_primary_term" : 2
}
```

데이터 입력과 조회 연습문제

- TourCompany의 고객관리를 위해 다음 데이터를 입력하십시오.
- Index 이름은 tourcompany로 사용

Doc Id	name	phone	holiday_dest	departure_date
1	Alfred	010-1234-5678	Disneyland	2017/01/20
2	Huey	010-2222-4444	Disneyland	2017/01/20
3	Naomi	010-3333-5555	Hawaii	2017/01/10
4	Andra	010-6666-7777	Bora Bora	2017/01/11
5	Paul	010-9999-8888	Hawaii	2017/01/10
6	Colin	010-5555-4444	Venice	2017/01/16

데이터 입력과 조회 연습문제

- 다음 임무를 수행하기 위해 쿼리문을 작성하고 데이터베이스에 적용하십시오.
  - BoraBora 여행은 공항테러 사태로 취소됐습니다. BoraBora 여행자의 명단을 삭제해주십시오.
  - Hawaii 단체 관람객의 요청으로 출발일이 조정됐습니다. 2017/01/10에 출발하는 Hawaii의 출발일을 2017/01/17일로 수정해주십시오.
  - 휴일 여행을 디즈니랜드로 떠나는 사람들의 핸드폰 번호를 조회하십시오.

배치 프로세스

_bulk API

- 여러 작업을 일괄적으로 수행 할 수 있는 기능
- 이 기능은 최대한 적은 네트워크 왕복으로 가능한 빠르게 여러 작업을 수행
- 다음 예제에서는 일괄 작업으로 두 개의 문서 (ID 1 - John Doe 및 ID 2 - Jane Doe)를 인덱싱

_bulk API 예제

- HTTP 바디 부분 끝에 반드시 엔터 추가 입력 필요

```
POST /customer/_bulk?pretty
{"index":{"_id":"1"}}
{"name": "John Doe" }
{"index":{"_id":"2"}}
{"name": "Jane Doe" }
[엔터]
```

- 다음 예는 첫 번째 문서 (ID 1)를 업데이트 한 다음 두 번째 문서 (ID 2)를 일괄 작업에서 삭제

```
POST /customer/_bulk?pretty
{"update":{"_id":"1"}}
{"doc": { "name": "John Doe becomes Jane Doe" } }
{"delete":{"_id":"2"}}
[엔터]
```

_bulk API

- Bulk API는 작업 중 하나에서 실패해도 전체 기능을 중지하지 않음
- 대량 API가 반환되면 각 액션에 대한 상태가 전송된 순서대로 제공
- 특정 액션이 실패했는지 여부를 확인 가능

```
1 { "create" : { "_index": "books", "_type": "book"} }
2 { "title": "The Tempest", "author": "William Shakespeare", "category":"Comedies", "written": "1610-03-01T11:34:00",
  "pages" : 62, "sell" : 275600000, "plot": "Magician Prospero, rightful Duke of Milan, and his daughter, Miranda,
have been stranded for twelve years on an island after Prospero's jealous brother Antonio (aided by Alonso, the
King of Naples) deposed him and set him adrift with the then-3-year-old Miranda. Gonzalo, the King's counsellor,
had secretly supplied their boat with plenty of food, water, clothes and the most-prized books from Prospero's
library. Possessing magic powers due to his great learning, Prospero is reluctantly served by a spirit, Ariel,
whom Prospero had rescued from a tree in which he had been trapped by the witch Sycorax. Prospero maintains
Ariel's loyalty by repeatedly promising to release the \"airy spirit\" from servitude. Sycorax had been banished
to the island, and had died before Prospero's arrival. Her son, Caliban, a deformed monster and the only
non-spiritual inhabitant before the arrival of Prospero, was initially adopted and raised by him. He taught
Prospero how to survive on the island, while Prospero and Miranda taught Caliban religion and their own language.
Following Caliban's attempted rape of Miranda, he had been compelled by Prospero to serve as the magician's slave.
In slavery, Caliban has come to view Prospero as a usurper and has grown to resent him and his daughter. Prospero
and Miranda in turn view Caliban with contempt and disgust."}
```

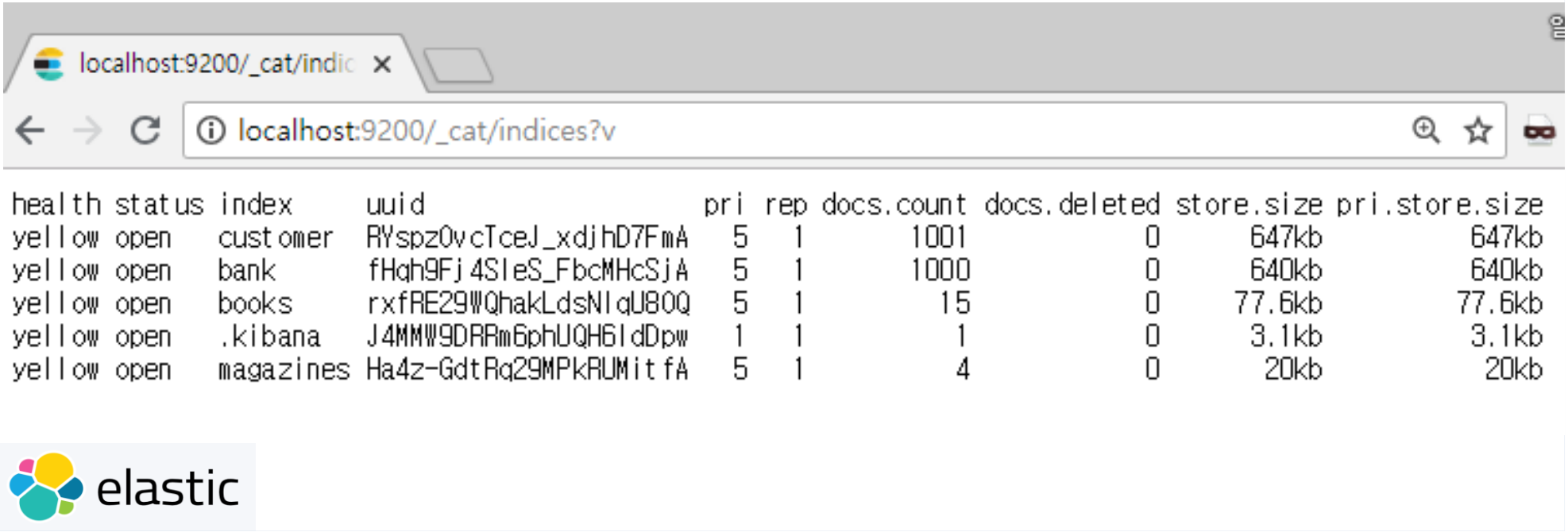
업로드할 데이터 확인

- 샘플 데이터셋

```
{
        "account_number": 0,
        "balance": 16623,
        "firstname": "Bradshaw",
        "lastname": "Mckenzie",
        "age": 29,
        "gender": "F",
        "address": "244 Columbus Place",
        "employer": "Euron",
        "email": "bradshawmckenzie@euron.com",
        "city": "Hobucken",
        "state": "CO"
}
```

데이터셋 로드하기

- 샘플 데이터 세트(accounts.json)를 클러스터에 로드
- 리눅스 curl 명령어
  - curl -H 'Content-Type: application/x-ndjson' -XPOST 'localhost:9200/bank/account/_bulk?pretty' --data-binary @accounts.json



elastic

검색 API

검색 시작하기

- 검색 용 REST API는 _search 엔드 포인트에서 액세스
- 엘라스틱서치는 쿼리에 사용하는 JSON 스타일 도메인 관련 언어 Query DSL을 제공

- 검색을 실행하는 기본적인 두 가지 방법
  - URI: 단순한 방법이 필요한 경우에는 “URI” 방식을 사용
  - 본문: 표현력을 높여 더 많은 정보를 전달하려면 “본문” 방식으로 JSON을 사용

검색 API

bank 인덱스의 모든 문서를 반환하는 URI 예제

```
GET /bank/_search?q=*&sort=account_number:asc&pretty
```

- bank 인덱스에서
- q=* 매개 변수는 Elasticsearch가 인덱스의 모든 문서와 일치하도록 지시
- sort=account_number:asc는 각 문서의 account_number 필드를 사용하여 결과를 오름차순으로 정렬
- pretty 매개 변수는 다시 Elasticsearch에게 예쁜 JSON 결과를 반환하도록 지시

검색 API

검색 API 결과 확인하기

항목	설명
took	Elasticsearch가 검색을 실행하는 데 걸린 시간 (밀리 초)
timed_out	검색 시간이 초과되었는지 여부를 알림
_shards	검색된 파편의 수와 성공 / 실패한 파편의 수를 알림
hits	검색 결과
hits.total	검색 조건과 일치하는 총 문서 수
hits.hits	검색 결과의 실제 배열 (기본값은 처음 10 개)
hits.sort	결과 정렬 키 (점수순 정렬시 누락)


```
{
    "took": 147,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "failed": 0
    },
    "hits": {
        "total": 1000,
        "max_score": null,
        "hits": [
            {
                "_index": "bank",
                "_type": "type1",
                "_id": "0",
                "_score": null,
                "_source": {
                    "account_number": 0,
                    "balance": 16623,
                    "firstname": "Bradshaw",
                    "lastname": "Mckenzie",
                    "age": 29,
                    "gender": "F",
                    "address": "244 Columbus Place",
                    "employer": "Euron",
                    "email": "bradshawmckenzie@euron.com",
                    "city": "Hobucken",
                    "state": "CO"
                },
                "sort": [
                    0
                ]
            },
            {
```

본문 메소드 요청 방법

- 본문 메소드를 사용하여 동일한 검색을 실행

```
POST /bank/_search
{
  "query": { "match_all": {} },
  "sort": [
    { "account_number": "asc" }
  ]
}
```

- 차이점
  - URI에 q=*를 전달하는 대신 _search API에 JSON 스타일 쿼리 요청 본문을 제공

검색 API

bank 인덱스에서 전체 검색하기

- query 에 match_all을 전달하면 조건없이 모든 도큐먼트를 검색

```
POST /bank/_search
{
        "query": { "match_all": {} }
}
```

검색 사이즈 정하기

- size 속성을 사용해 쿼리되는 데이터의 크기를 지정
- from 속성을 사용해 쿼리되는 데이터의 위치를 지정

```
POST /bank/_search
{
        "query": { "match_all": {} },
        "size": 1
}
```

결과를 한 개만 검색
(Default: 10)

```
POST /bank/_search
{
        "query": { "match_all": {} },
        "from": 10,
        "size": 10
}
```

결과의 10번째 항목부터
(Default:1)

검색 API

검색 결과에 대한 정렬을 수행

- sort 속성에 특정 필드(balance)를 기준으로 내림차순으로 정렬하고 상위 10개의 문서를 반환

```
GET /bank/_search
{
  "query": { "match_all": {} },
  "sort": { "balance": { "order": "desc" } }
}
```

특정 필드 내용만 검색

- 검색 조회의 _source 필드에서 원하는 필드만 지정해서 반환

```
GET /bank/_search
{
  "query": { "match_all": {} },
  "_source": ["account_number", "balance"]
}
```

특정 필드에 문자가 일치하는 도큐먼트 검색

- 20이라는 번호가 매겨진 account_number를 반환

```
GET /bank/_search
{
  "query": { "match": { "account_number": 20 } }
}
```

- Address가 mill과 일치하면 반환

```
GET /bank/_search
{
  "query": { "match": { "address": "mill" } }
}
```

검색 API

특정 필드에 문자열이 일치하는 도큐먼트 검색

- Address가 mill 또는 lane과 일치하면 반환

```
POST /bank/_search
{
  "query": { "match": { "address": "mill lane" } }
}
```

- Address에 “mill lane”이라는 문구가 일치하면 반환

```
POST /bank/_search
{
  "query": { "match_phrase": { "address": "mill lane" } }
}
```

둘 이상의 조건이 일치하는 검색

- 두 개의 일치 쿼리를 작성하고 주소에 "mill"및 "lane"을 포함하는 모든 계정을 반환

```
POST /bank/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "address": "mill" } },
        { "match": { "address": "lane" } }
      ]
    }
  }
}
```

부울 쿼리	사용하는 컨텍스트	점수에 영향을 미치는가?	결과가 매칭되면 출력되는가?	정확히 매칭되는가?
must	Query	O	O	O
filter	Filter	X	O	O
should	Query	O	O	X
must_not	Filter	X	O	O

검색 연습하기

- 검색 결과에는 어떤 것들이 나올지 추측하고 검색해보자.

```
GET /bank/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "age": "40" } }
      ],
      "must_not": [
        { "match": { "state": "ID" } }
      ]
    }
  }
}
```

검색 API

데이터 검색

- 데이터 검색을 위한 데이터베이스 업데이트
- curl -H 'Content-Type: application/x-ndjson' -XPOST 'localhost:9200/_bulk?pretty' --data-binary @books.json
- curl -H 'Content-Type: application/x-ndjson' -XPOST 'localhost:9200/_bulk?pretty' --data-binary @magazines.json

```
{ "create" : { "_index": "books", "_type": "book", "_id": 1} }
{ "title": "The Tempest", "author": "William Shakespeare", "category":"Comedies", "written": "16
{ "create" : { "_index": "books", "_type": "book", "_id": 2} }
{ "title": "The Merchant of Venice", "author": "William Shakespeare", "category":"Comedies", "wr
{ "create" : { "_index": "books", "_type": "book", "_id": 3} }
{ "title": "Romeo and Juliet", "author": "William Shakespeare", "category":"Tragedies", "written
{ "create" : { "_index": "books", "_type": "book", "_id": 4} }
{ "title": "King Lear", "author": "William Shakespeare", "category":"Tragedies", "written": "160
{ "create" : { "_index": "books", "_type": "book", "_id": 5} }
{ "title": "Hamlet", "author": "William Shakespeare", "category":"Tragedies", "written": "1599-0
{ "create" : { "_index": "books", "_type": "book", "_id": 6} }
{ "title": "Othello", "author": "William Shakespeare", "category":"Tragedies", "written": "1603-
{ "create" : { "_index": "books", "_type": "book", "_id": 7} }
{ "title": "The Adventures of Tom Sawyer", "author": "Mark Twain", "category":"Folk", "written":
{ "create" : { "_index": "books", "_type": "book", "_id": 8} }
{ "title": "The Prince and the Pauper", "author": "Mark Twain", "category":"Children's literatur
```

elastic

검색 API

검색(_search) API

- 엘라스틱서치에서의 검색은 인덱스 또는 타입 단위로 수행
- _search API 사용
- 질의는 q 매개변수의 값으로 입력
- hamlet이라는 검색어로 검색

질의	질의문
books 인덱스에서 hamlet 검색	localhost:9200/**books/_search?q=hamlet**
전체 인덱스에서 time 검색	localhost:9200/**_search?q=time**

검색 API

_search API 검색 결과

took에 응답시간 기록 밀리초로 표시
("took" : 7 → 0.007초)

각 검색 결과에 대한 점수 표시

hits에 결과 표시

GET localhost:9200/books/book/_search?q=hamlet Params

Body Cookies Headers (3) Tests Status: 20

Pretty Raw Preview JSON


```
{
  "took": 7,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 2.161416,
    "hits": [
      {
        "_index": "books",
        "_type": "book",
        "_id": "AVw9yZdtB1BdYSeNM7P1",
        "_score": 2.161416,
        "_source": {
          "id": 5,
          "title": "Hamlet",
          "author": "William Shakespeare",
          "category": "Tragedies",
          "written": "1599-06-01T12:34:00",
          "pages": 172,
          "sell": 146100000,
          "plot": "The protagonist of Hamlet is Prince Hamlet of Denmark, son of the recently
              nephew of King Claudius, his father's brother and successor. Claudius hastily ma
              Gertrude, Hamlet's mother. Denmark has a long-standing feud with neighbouring No
              the Norwegian prince, Fortinbras, is expected."
        }
      }
    ]
  }
}
```


elastic

검색 API

특정 필드 검색

- q 매개변수에 <필드명: 질의>입력

질의	질의문
전체 인덱스의 title 필드에서 time 검색	/_search?q=**title:time**

다중 조건 검색

- and와 or를 사용하여 다수의 조건을 검색

질의	질의문
title 필드에서 time과 machine을 검색	/_search?q=title:time **AND** machine

elastic

explain : 점수 계산에 사용된 상세 값 출력

질의	질의문
explain 매개변수를 사용해서 검색 처리 결과 표시	/_search?q=title:time**&explain**

요약된 전체 hit 수와 점수(score) 등의 메타 정보를 출력

- _source false로 설정하면
- 도큐먼트 출력 생략

질의	질의문
_source 매개변수를 false로 설정해 도큐먼트 내용을 배제하고 검색	/_search?q=title:time**&_source=false**

검색 API

출력 결과에 표시할 필드를 지정

- _source에 표시할 필드를 쉼표(,)로 구분하여 입력

질의	질의문
title, author, category 필드만 출력	/_search?q=title:time**&_source=title,author,category**

검색 결과의 출력 순서 정렬

- sort=필드명 형식 사용 (디폴트로 _score 값 기준)
- 내림차순 정렬 : sort=필드명:desc (디폴트로 asc(오름차순))

질의	질의문
pages 필드를 기준으로 오름차순 정렬	/_search?q=author:jules**&sort=pages**
pages 필드를 기준으로 내림차순 정렬	/_search?q=author:jules**&sort=pages:desc**

검색 API 연습문제

- TourCompany에입력했던 데이터가 모두 날아갔다. 이런 상황을 미리 방지하기 위해 벌크 데이터를 만들고 API를 사용하여 업로드 해보자.
- Index 이름은 tourcompany로 한다.

Doc Id	name	phone	holiday_dest	departure_date
1	Alfred	010-1234-5678	Disneyland	2017/01/20
2	Huey	010-2222-4444	Disneyland	2017/01/20
3	Naomi	010-3333-5555	Hawaii	2017/01/10
4	Andra	010-6666-7777	Borabora	2017/01/11
5	Paul	010-9999-8888	Hawaii	2017/01/10
6	Colin	010-5555-4444	Venice	2017/01/16

검색 API 연습문제

- 좀더 효과적인 임무 수행을 위해 검색 기능을 수행하는 쿼리를 작성하십시오.

1. tourcompany 인덱스에서 010-3333-5555를 검색하십시오.
2. 휴일 여행을 디즈니랜드로 떠나는 사람들의 핸드폰 번호를 조회하십시오(phone 필드만 출력).
3. departure date가 2017/01/10과 2017/01/11인 사람을 조회하고 이름 순으로 출력하십시오. (name과 departure date 필드만 출력)
4. BoraBora 여행은 공항테러 사태로 취소됐습니다. BoraBora 여행자의 명단을 삭제해주십시오.
5. Hawaii 단체 관람객의 요청으로 출발일이 조정됐습니다. 2017/01/10에 출발하는 Hawaii의 출발일을 2017/01/17일로 수정해주십시오.

검색한 결과를 업데이트하는 API 예제

```
POST my-index-000001/_update_by_query
{
  "script": {
    "source": "ctx._source.count++", "lang": "painless"
  },
  "query": {
    "term": { "user.id": "kimchy" }
  }
}
```

해답

1. tourcompany 인덱스에서 010-3333-5555를 검색하십시오.
   - GET /tourcompany/_search?q="010-3333-5555"
2. 휴일 여행을 디즈니랜드로 떠나는 사람들의 핸드폰 번호를 조회하십시오(phone 필드만 출력).
   - GET /tourcompany/_search?q=holiday_dest: Disneyland&_source=phone,holiday_dest
3. departure date가 2017/01/10과 2017/01/11인 사람을 조회하고 이름 순으로 출력하십시오. (name과 departure date 필드만 출력)
   - 127.0.0.1:9200/tourcompany/_search?q=departure_date:"2017/01/10" or departure_date:"2017/01/11"&_source=name,phone,holiday_dest&sort=name.keyword:asc

해답

4. BoraBora 여행은 공항테러 사태로 취소됐습니다. BoraBora 여행자의 명단을 삭제해주십시오.

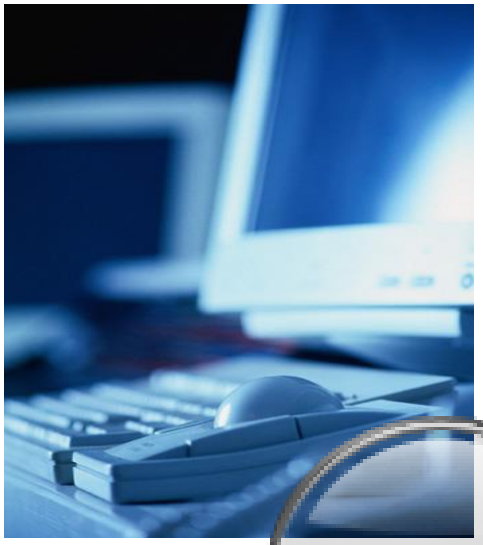
```
POST /tourcompany/_update_by_query
{
  "script": {"source": "ctx.op='delete'", "lang": "painless" },
  "query": {"match": { "holiday_dest": "Bora Bora" } }
}
```

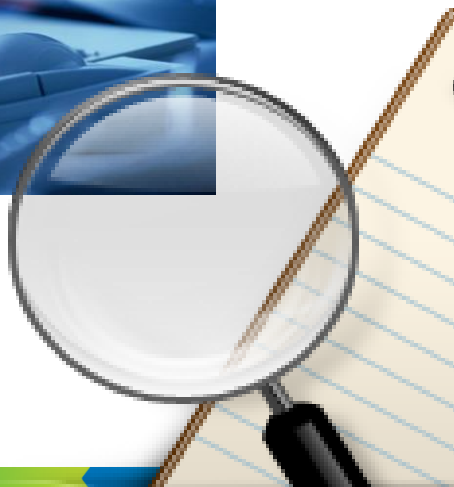
5. Hawaii 단체 관람객의 요청으로 출발일이 조정됐습니다. 2017/01/10에 출발하는 Hawaii의 출발일을 2017/01/17일로 수정해주십시오.

```
POST tourcompany/_update_by_query
{
  "script": {"source": "ctx._source.departure_date='2017/01/17'", "lang": "painless"},
  "query": {
    "bool": {
      "must": [
        {"match": {"departure_date": "2017/01/10"}},
        {"match": {"holyday_dest": "Hawaii"}}
        ]
    }
  }
}
```

대시보드 시각화를 위한 KIBANA 기초

- KIBANA 소개와 설치
- KIBANA 데이터 준비
- KIBANA 시각화





KIBANA 소개와 설치

키바나 소개

- Elasticsearch와 함께 작동하도록 설계된 오픈 소스 분석 및 시각화 플랫폼
- Elasticsearch 색인에 저장된 데이터를 검색,보기 및 상호 작용
- 고급 데이터 분석을 쉽게 수행하고 다양한 차트, 테이블 및 맵에서 데이터를 시각화
- 간단한 브라우저 기반의 인터페이스를 통해 실시간으로 Elasticsearch 쿼리의 변경 사항을 표시하는 동적 대시 보드를 신속하게 만들고 공유
- 간단한 설치!

elastic

KIBANA 소개와 설치

도커로 키바나 설치하기

```
sudo apt update && sudo apt install -y docker.io
docker network create elastic
docker run -d  - name es01-test  - net elastic  - p 9200:9200  - p 9300:9300  - e
" discovery.type=single-node "  elasticsearch:7.14.1
Docker run  - d  - name kib01-test  - net elastic  - p 5601:5601  - e
"ELASTICSEARCH_HOSTS=http://es01-test:9200" kibana:7.14.1
```

엘라스틱서치 버전

- Kibana는 동일한 버전의 Elasticsearch 노드에 대해 실행되도록 구성되어야 함
- 공식적으로 지원되는 구성

엘라스틱서치 도메인 지정

- 환경변수 ELASTICSEARCH_HOSTS에 엘라스틱서치 도메인 설정하여 시작

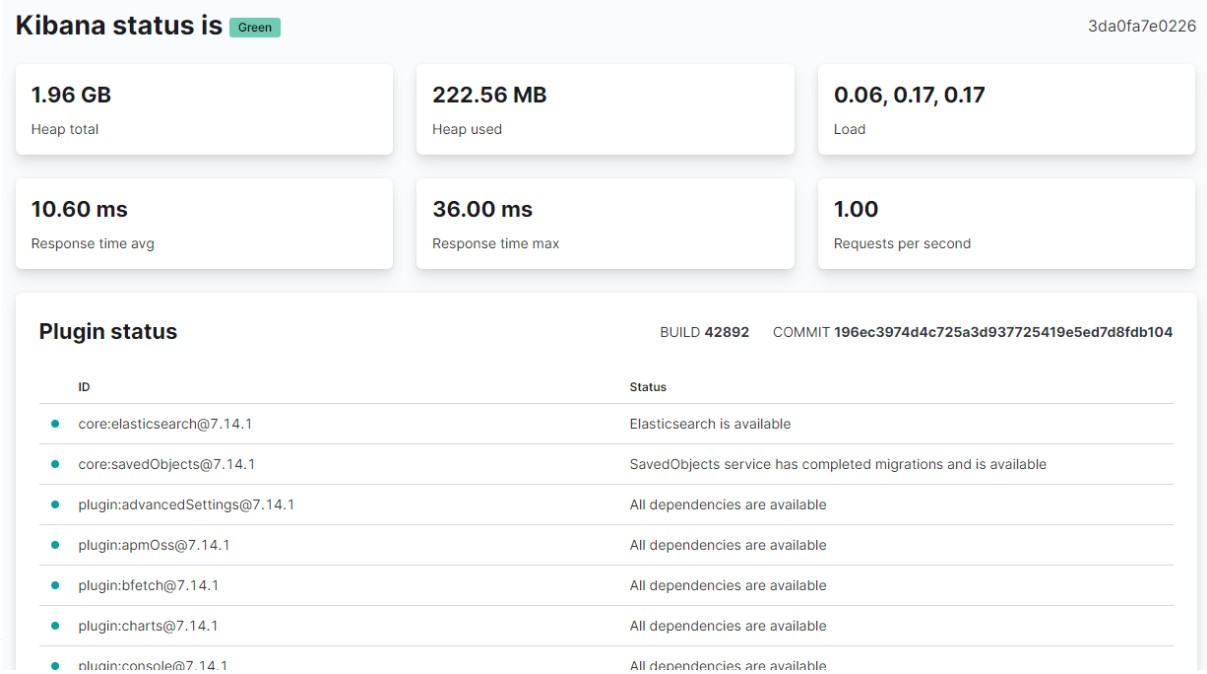
윈도우에 설치된 .zip 아카이브의 디렉토리 레이아웃

유형	설명	기본 경로
home	Kibana 홈 디렉토리 ($KIBANA_HOME)	아카이브 압축을 풀어 생성된 디렉토리
bin	Kibana 서버를 시작하기 kibana 파일과 플러그인 설치를 위한 kibana-plugin을 포함한 바이너리 스크립트 경로	$KIBANA_HOME\bin
config	kibana.yml을 포함한 구성 파일	$KIBANA_HOME\config
data	Kibana 및 해당 플러그인이 디스크에 기록하는 데이터 파일의 위치 입니다.	$KIBANA_HOME\data
optimize	번역된 소스 코드 특정 관리 작업으로 인해 소스 코드가 즉석으로 변환	$KIBANA_HOME\optimize
plugins	플러그인 파일 위치, 각 플러그인은 하위 디렉토리에 포함	$KIBANA_HOME\plugins

elastic

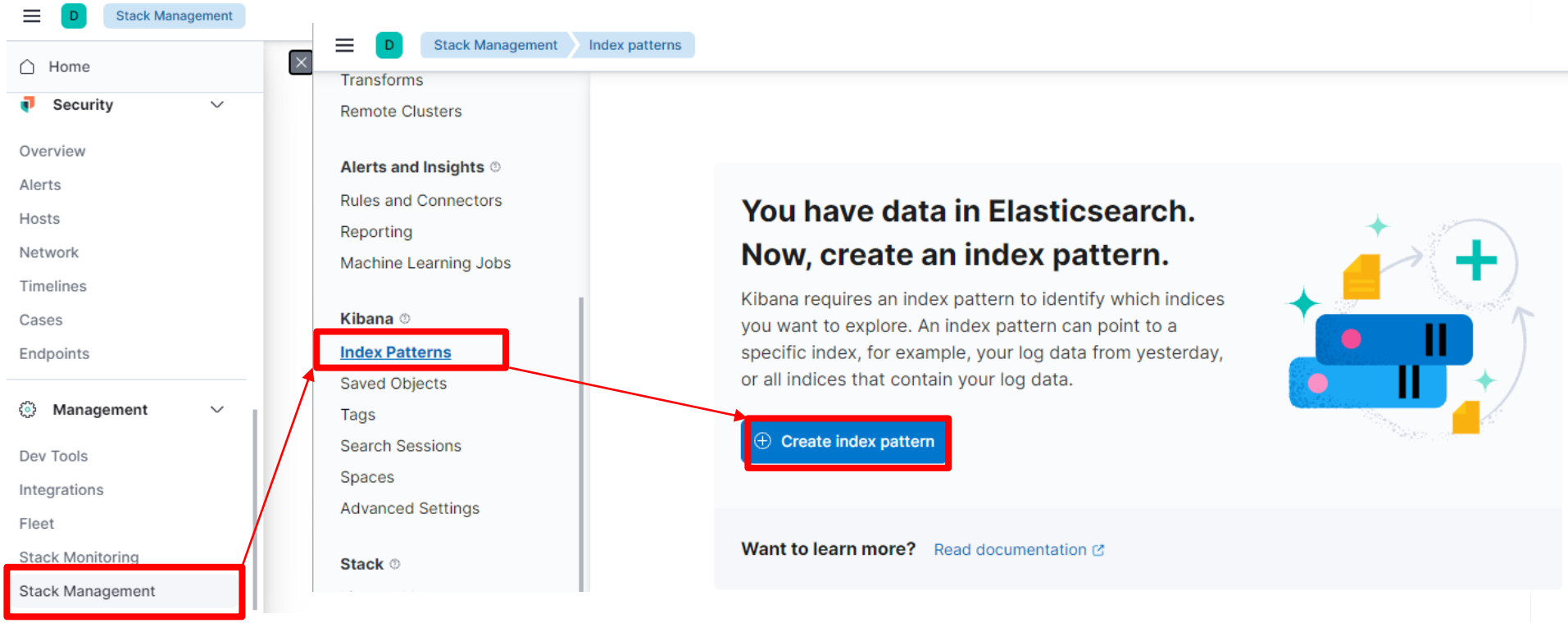
키바나 접속하기

- kibana는 포트 5601을 통해 액세스하는 웹 응용 프로그램
  - localhost : 5601 또는 http://<ip>:5601
- Kibana 서버의 상태 페이지
  - localhost:5601/status
  - 상태 페이지는 서버의 자원 사용에 대한 정보를 표시
  - 설치된 플러그인을 나열



엘라스틱서치 데이터 불러오기

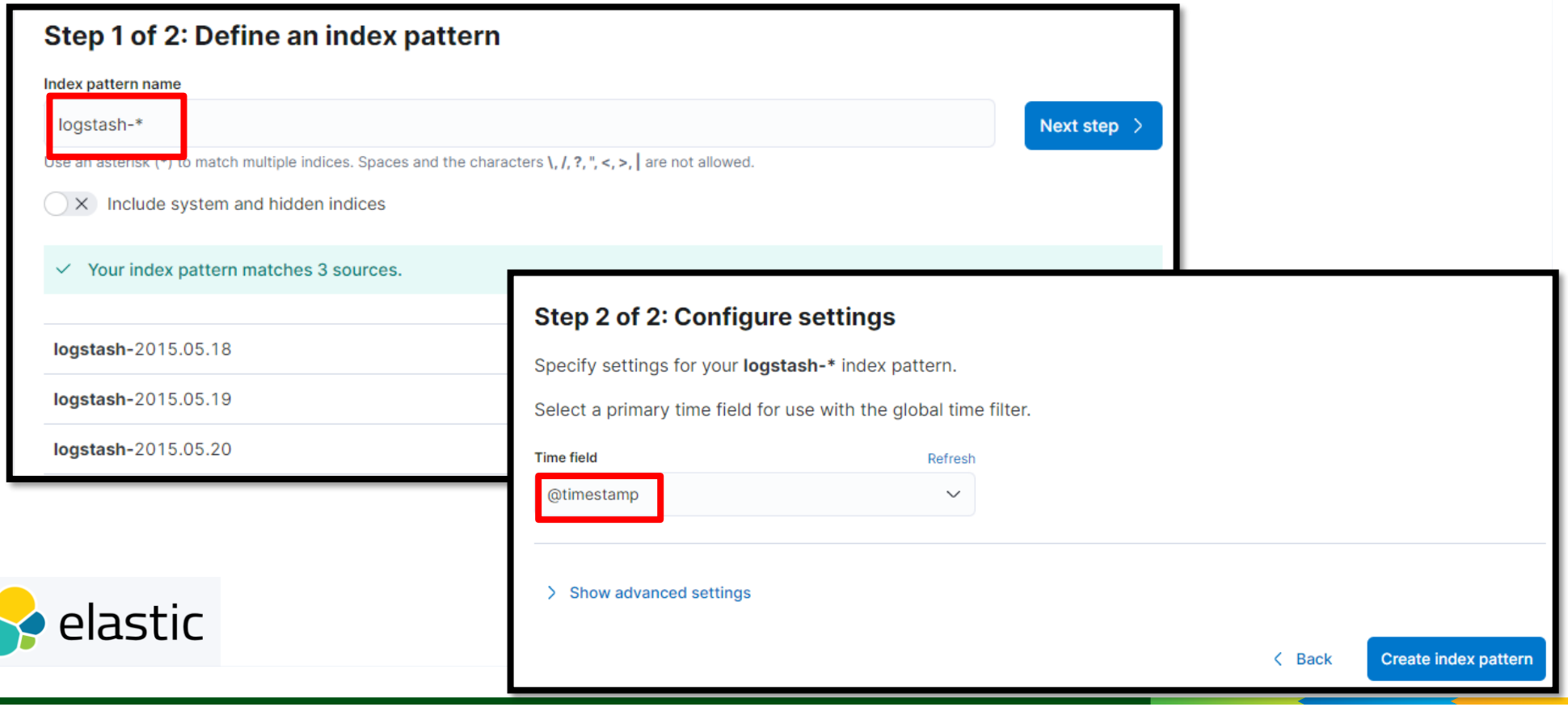
- 브라우저에서 Kibana UI에 액세스하려면 포트 5601로 접속
- Elasticsearch 색인 이름과 일치하는 색인 패턴을 지정



elastic

엘라스틱서치 데이터 불러오기

- 엘라스틱 서치에서 가져올 인덱스의 이름 패턴을 입력 logstash-*
- 시계열 데이터는 @타임스탬프로 결정



elastic

KIBANA 데이터 준비

Kibana 튜토리얼 순서

- Elasticsearch에 샘플 데이터 세트로드
- 인덱스 패턴 정의
- Discover를 사용하여 샘플 데이터 탐색
- 샘플 데이터의 시각화 설정
- 시각화를 대시 보드로 어셈블

Kibana 튜토리얼 준비하기

- 샘플데이터 업로드하기
  - 윌리엄 셰익스피어 (William Shakespeare)의 전체 작품은 적절하게 필드로 파싱 - shakespeare.json
  - 무작위로 생성 된 데이터로 구성된 가상 계정 집합 - accounts.json
  - 임의로 생성 된 로그 파일 세트 - logs.jsonl

shakespeare.json

```
{
 "line_id": INT,
 "play_name": "String",
 "speech_number": INT,
 "line_number": "String",
 "speaker": "String",
 "text_entry": "String",
}
```

logs.jsonl

```
{
 "memory": INT,
 "geo.coordinates": "geo_point",
 "@timestamp": "date"
}
```

accounts.json

```
{
 "account_number": INT,
 "balance": INT,
 "firstname": "String",
 "lastname": "String",
 "age": INT,
 "gender": "M or F",
 "address": "String",
 "employer": "String",
 "email": "String",
 "city": "String",
 "state": "String"
}
```

elastic

인덱스 매핑 설정

- 데이터를 로드하기 전에 매핑을 먼저 수행!
  - 매핑을 수행하지 않은 경우에는 임의의 데이터 형태로 매핑
- 매핑이란?
  - 인덱스의 문서를 논리적 그룹으로 나누고 필드의 검색 가능성 또는 토큰화되었는지 또는 별도의 단어로 분리되는지와 같은 필드의 특성을 지정
  - account 데이터 세트에는 매핑이 필요하지 않음

```
PUT /shakespeare
{
  "mappings" : {
    "properties" : {
      "speaker" : {"type": "keyword" },
      "play_name" : {"type": "keyword" },
      "line_id" : { "type" : "integer" },
      "speech_number" : { "type" : "integer" }
    }
  }
}
```

```
#18~20까지 세개의 인덱스 구성 필요
PUT /logstash-2015.05.18
{
  "mappings": {
    "properties": {
      "geo": {
        "properties": {
        "coordinates": { "type": "geo_point" }
        }
      }
    }
  }
}
```

elastic

엘라스틱서치의 데이터 형

- 벌크 데이터를 사용하여 Elasticsearch에 데이터 세트를 로드

```
PUT /shakespeare
{
  "mappings" : {
    "properties" : {
      "speaker" : {"type": "keyword" },
      "play_name" : {"type": "keyword" },
      "line_id" : { "type" : "integer" },
      "speech_number" : { "type" : "integer" }
…
```

```
#18~20까지 세개의 인덱스 구성 필요
PUT /logstash-2015.05.18
{
  "mappings": {
    "properties": {
      "geo": {
        "properties": {
        "coordinates": { "type": "geo_point" }
        }
…
```

keyword

키워드 필드는 분석되지 않음

단일 단위로 처리 (문자열은 여러 단어가 포함)

integer

정수형 데이터 타입

geo_point

위도 / 경도 쌍에 지리적 위치로 레이블을 지정

elastic

샘플 데이터 업로드

- 벌크 데이터를 사용하여 Elasticsearch에 데이터 세트를 로드

- curl -H 'Content-Type: application/x-ndjson' -XPOST 'localhost:9200/bank/_bulk?pretty' --data-binary @accounts.json

- curl -H 'Content-Type: application/x-ndjson' -XPOST 'localhost:9200/_bulk?pretty' --data-binary @shakespeare_6.0.json

- curl -H 'Content-Type: application/x-ndjson' -XPOST 'localhost:9200/_bulk?pretty' --data-binary @logs.jsonl

KIBANA 데이터 준비

다음 명령을 사용하여 성공적으로 로드되었는지 확인

- GET /_cat/indices?v

Pretty Raw Preview Text

```
health status index                 uuid                   pri rep docs.count docs.deleted store.size pri.store.size
yellow open   apache_elastic_example Gr33jX5_SM2xwfvgqVlHCg   5   1       3312            0        2mb            2mb
yellow open   logstash-2015.05.18   B_VNMbyATB2lLavCGstmfg   5   1       4631            0     29.8mb         29.8mb
yellow open   logstash-2015.05.19   SBrvUwAkTFW2GxAuL_8I2A   5   1       4624            0     30.3mb         30.3mb
yellow open   bank                  r3_dtMGcT8KIlJ7W1yw-mA   5   1       1000            0    647.9kb        647.9kb
yellow open   shakespeare           UEINXXSiR9apVrAQNSGGIA   5   1     111396            0     28.5mb         28.5mb
yellow open   logstash-2015.05.20   8YeYD1N_RDWCv-xZt1BsiQ   5   1       4750            0       30mb           30mb
yellow open   .kibana               a-QQKs6iTzmvANLv8lO9LA   1   1          2            0     11.8kb         11.8kb
```

elastic

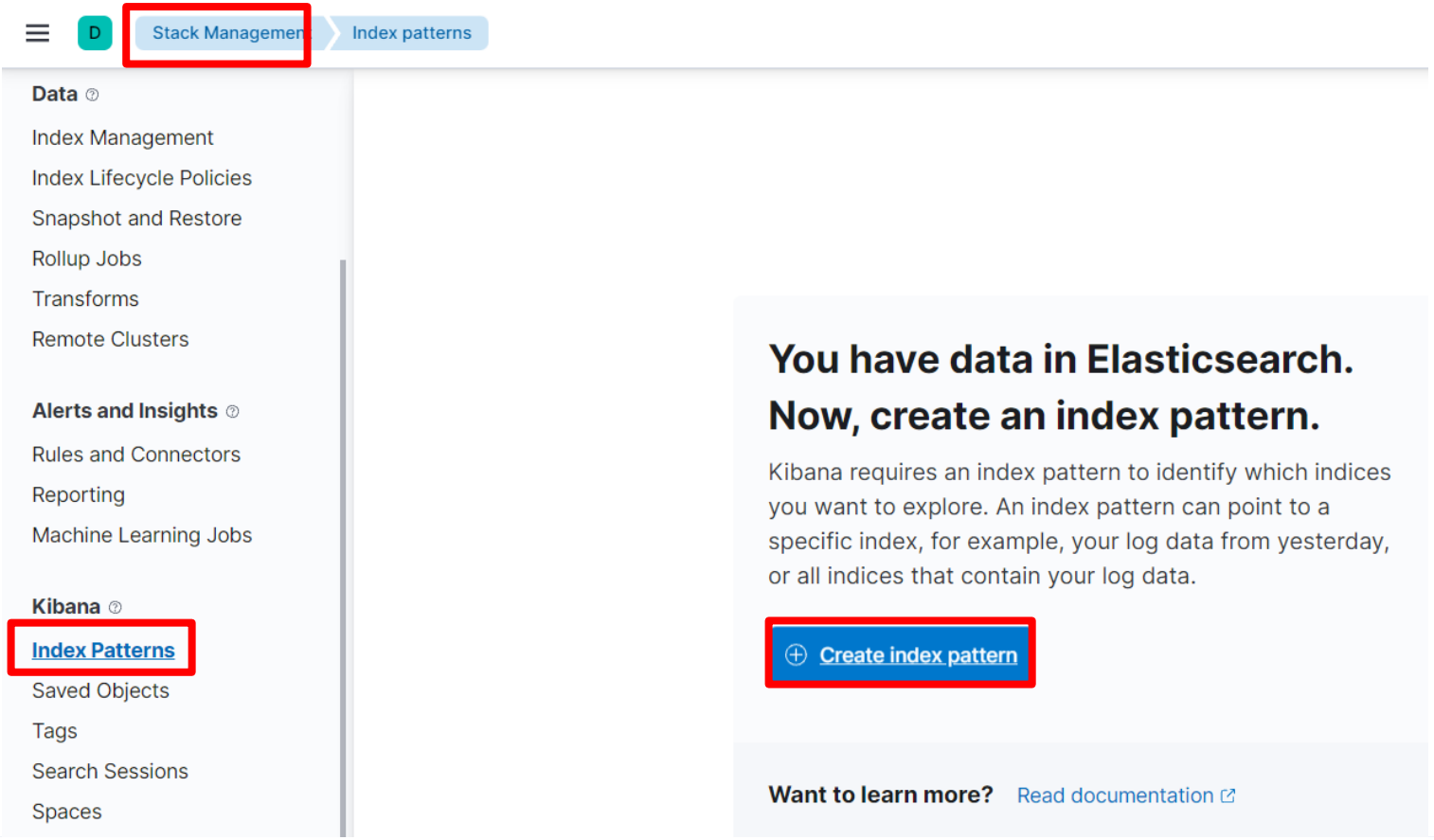
Index 패턴 정의하기

- Elasticsearch에 로드된 각 데이터 세트에는 인덱스 패턴 존재
  - shakespeare 데이터 세트 : shakespeare라는 인덱스(shakes*)
  - account 데이터 세트 : bank라는 인덱스(ba*)
  - logs 데이터 세트 : YYYY.MM.DD 형식의 날짜가 포함(5월에 대한 색인 패턴,logstash-2015.05*)

1. **shakespeare와 account 데이터세트에는 시계열 데이터를 포함**
2. **데이터 세트에 대한 인덱스 패턴이 시간 기반 이벤트가 포함되어 있는지 확인 필요**

Index 패턴 정의하기

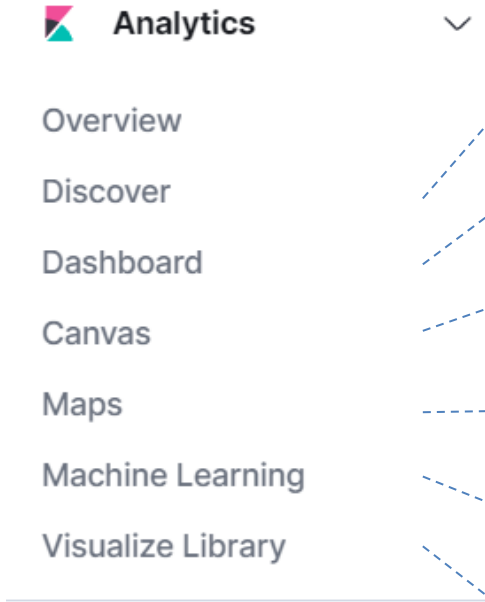
- Management 탭을 클릭 한 다음 Index Patterns 탭을 클릭
- +Add New 새로 추가를 클릭하여 새 인덱스 패턴을 정의



KIBANA 시각화

KIBANA 시각화

키바나 메뉴



데이터베이스를 탐색하고 검색하는 기능

각종 차트를 원하는대로 배치하는 대시보드 기능

CSS 등을 사용해 자신만의 스타일을 꾸밀 수 있는 기능

데이터를 지도에 표현할 수 있는 기능

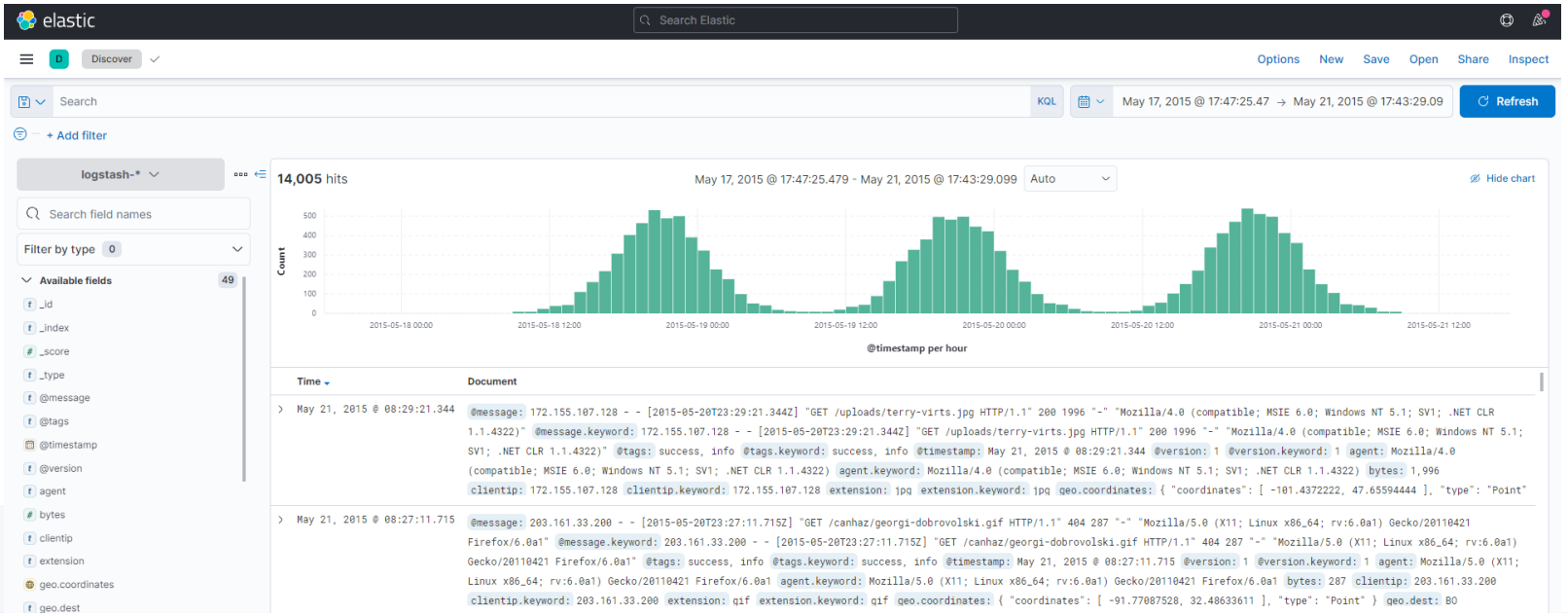
머신러닝 기능을 사용해 이상 탐지하는 기능 (30일 trial)

데이터로 차트를 그리고 저장할 수 있는 차트 저장소

elastic

Kibana 시각화: Discover

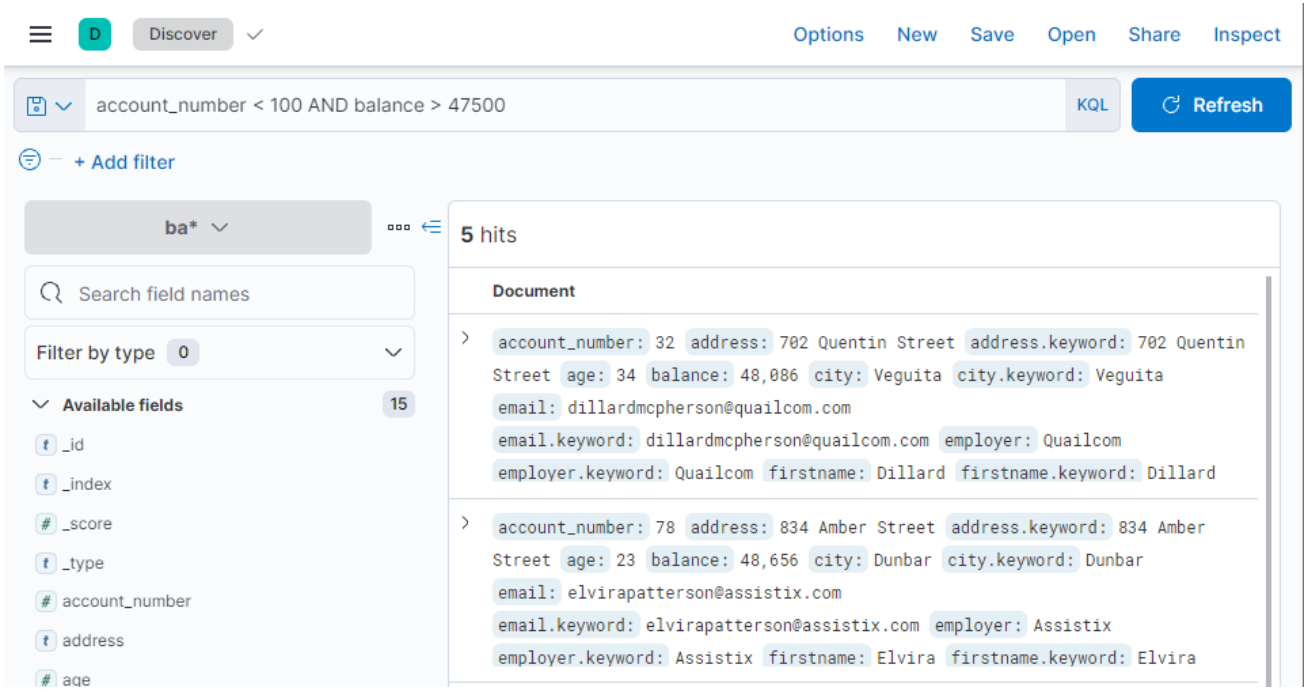
- 결과를 탐색하고 Visualize에서 저장된 검색의 시각화
- 쿼리 바에서 Elasticsearch query를 입력하여 데이터를 검색
  - 관심있는 필드 이름과 값을 사용하여 검색을 구성
  - 숫자 필드에서는 큼 (>), 작음 (<) 또는 같음 (=)과 같은 비교 연산자 지원
  - 논리 연산자 AND, OR 및 NOT 등 지원



Kibana 시각화: Discover

- ba* 인덱스를 선택하고 다음 쿼리를 실행
  - 잔액 : 47,500을 초과하는
  - 0에서 99 사이의 모든 계좌 번호
  - 샘플 뱅크 데이터를 검색 할 때 계정 번호 8, 32, 78, 85 및 97의 5 개의 결과가 반환됩니다.

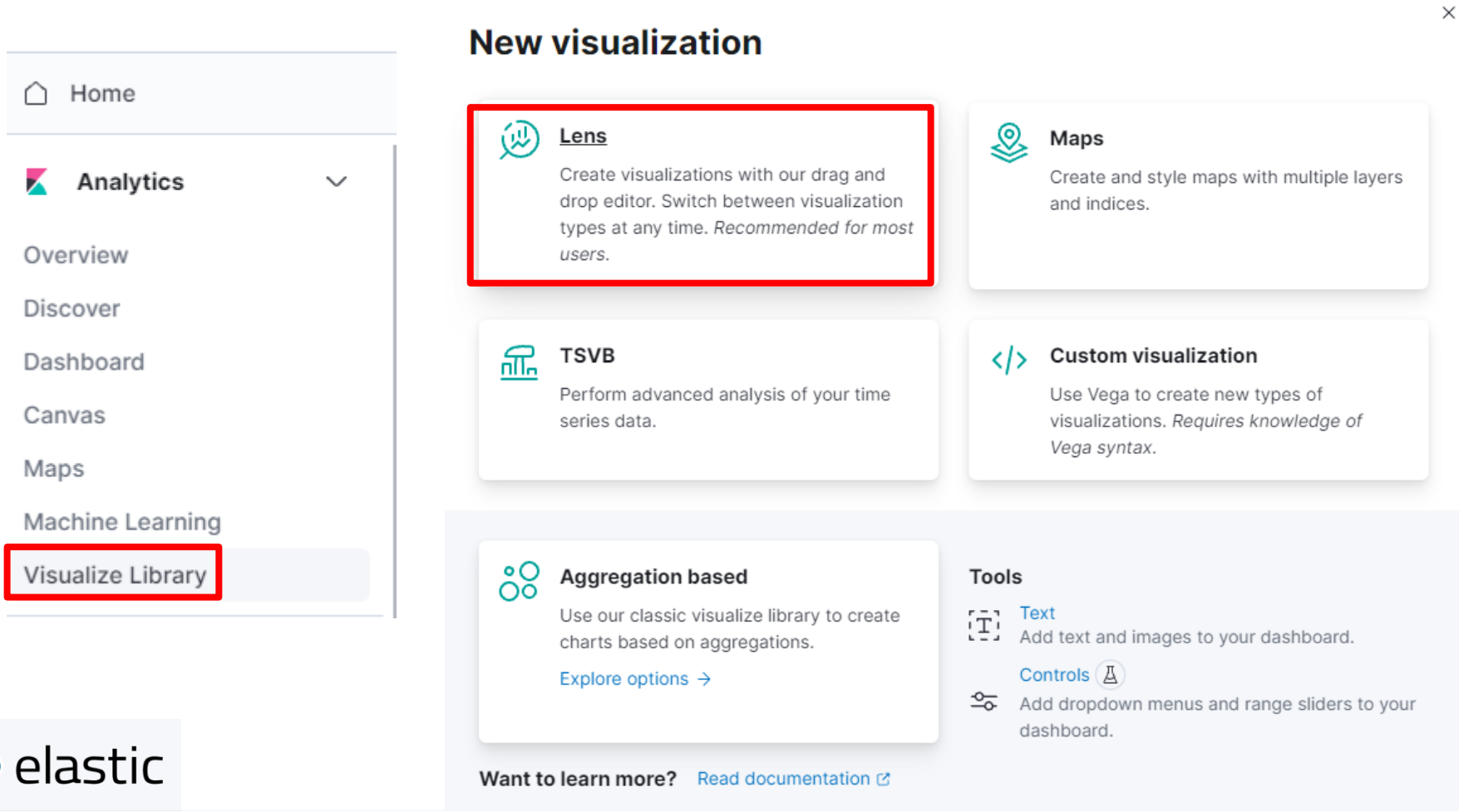
```
account_number < 100
AND balance > 47500
```



elastic

Kibana 시각화: Visualize Library

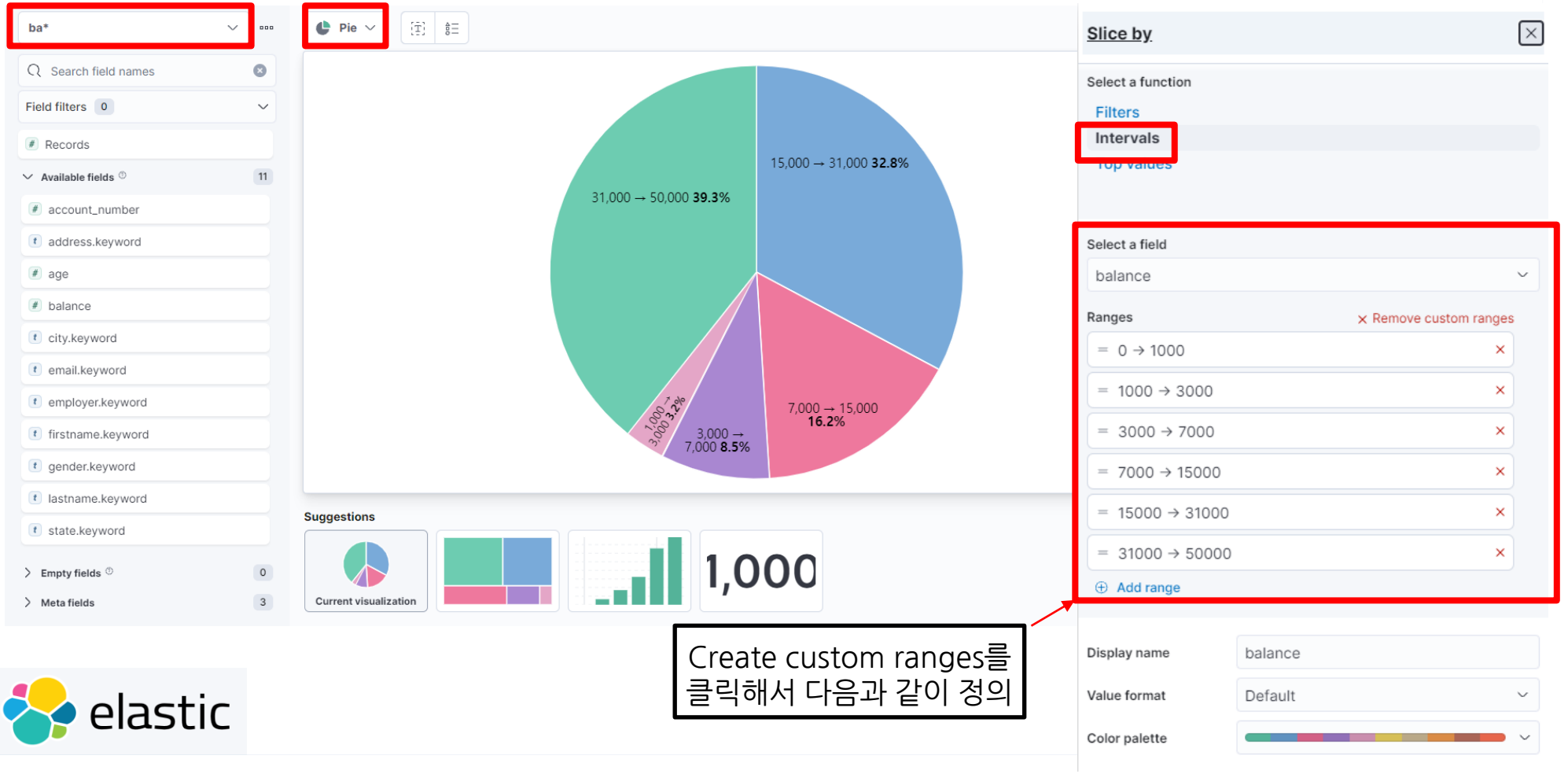
- 여러 가지 방법으로 데이터를 시각화



elastic

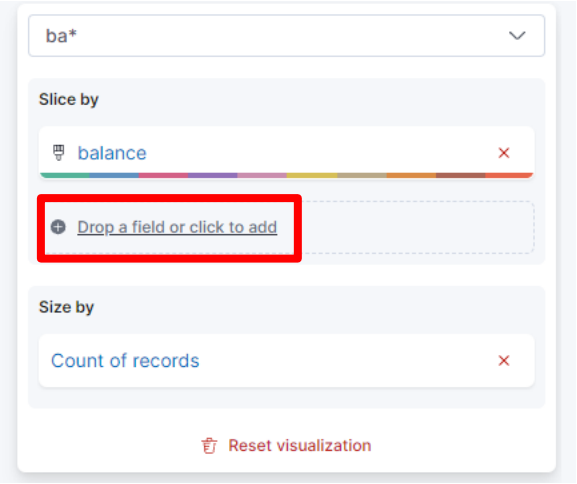
Kibana 시각화: Visualize

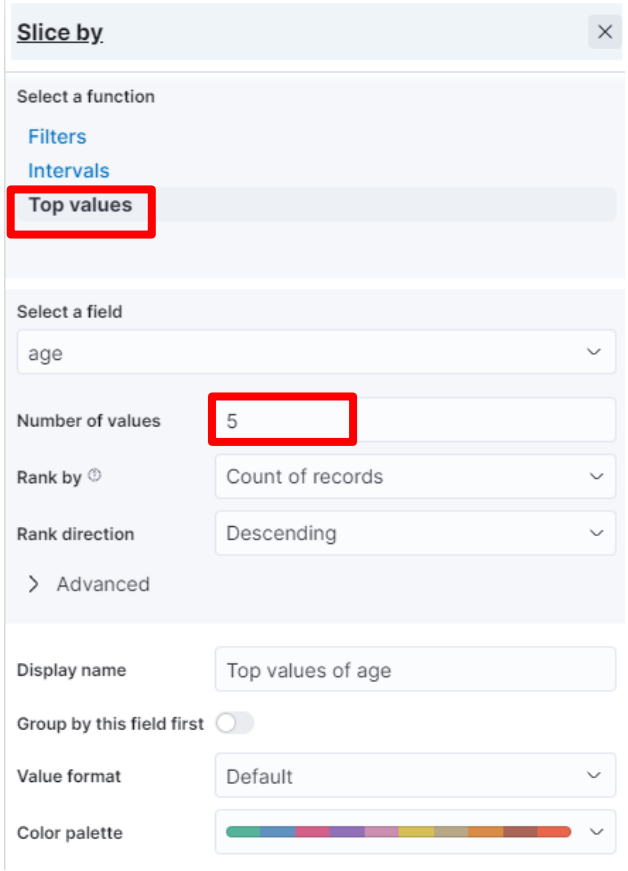
- 계좌 잔액에 대한 통계 정보 만들기 - Pie 차트를 사용, ba* 인덱스 선택

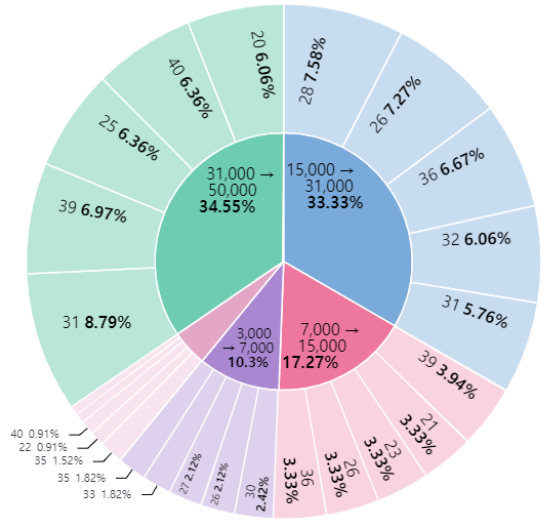


Kibana 시각화: Visualize

- 파이 차트에 “age 변수” 추가



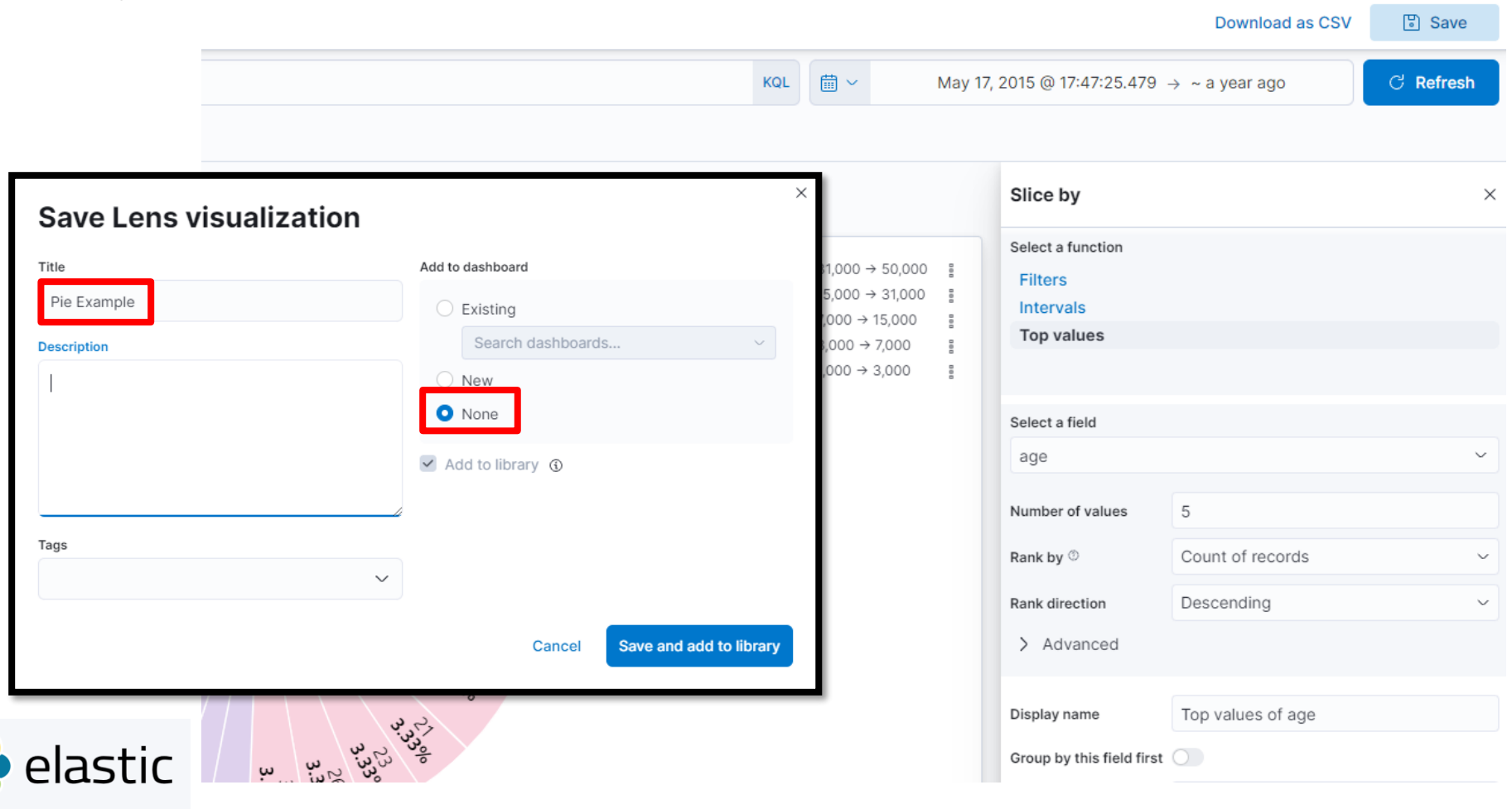




elastic

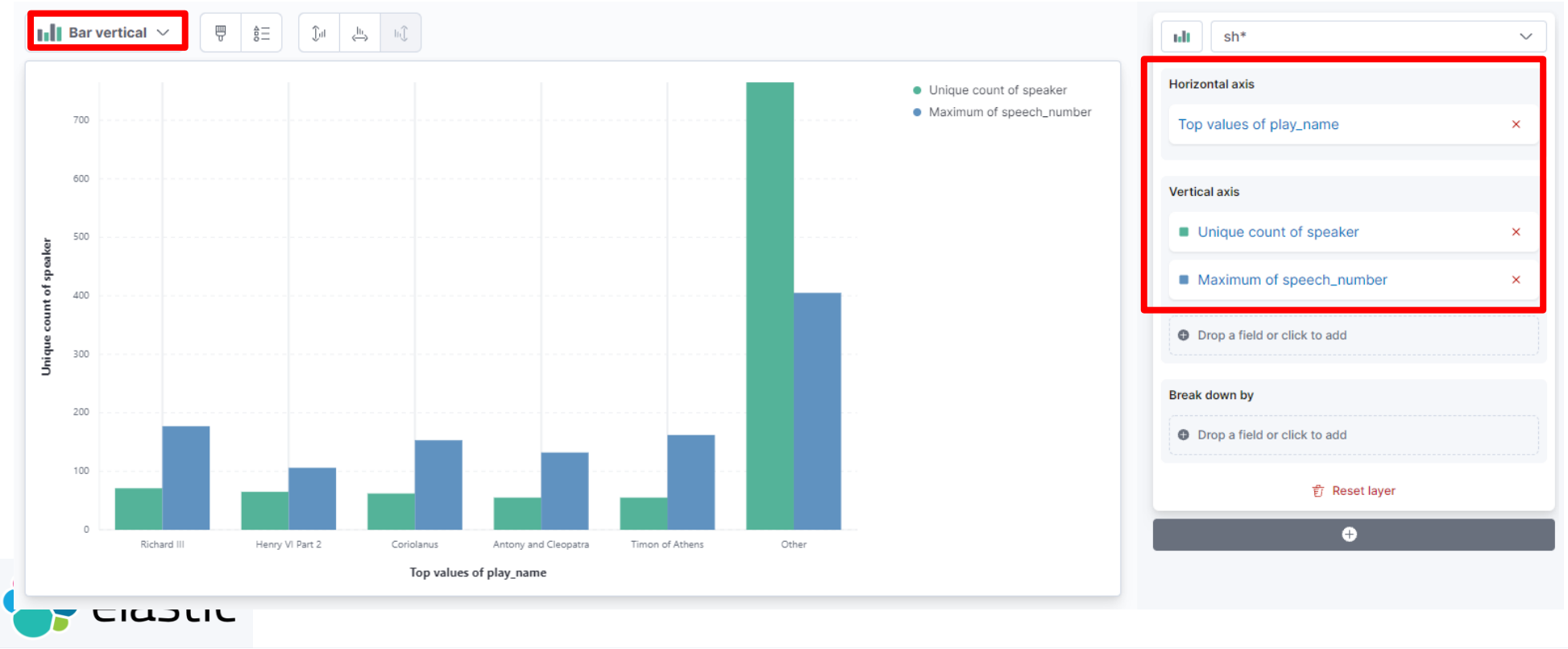
Kibana 시각화: Visualize

- 상단에 Save 버튼을 누르고 Pie Example로 저장



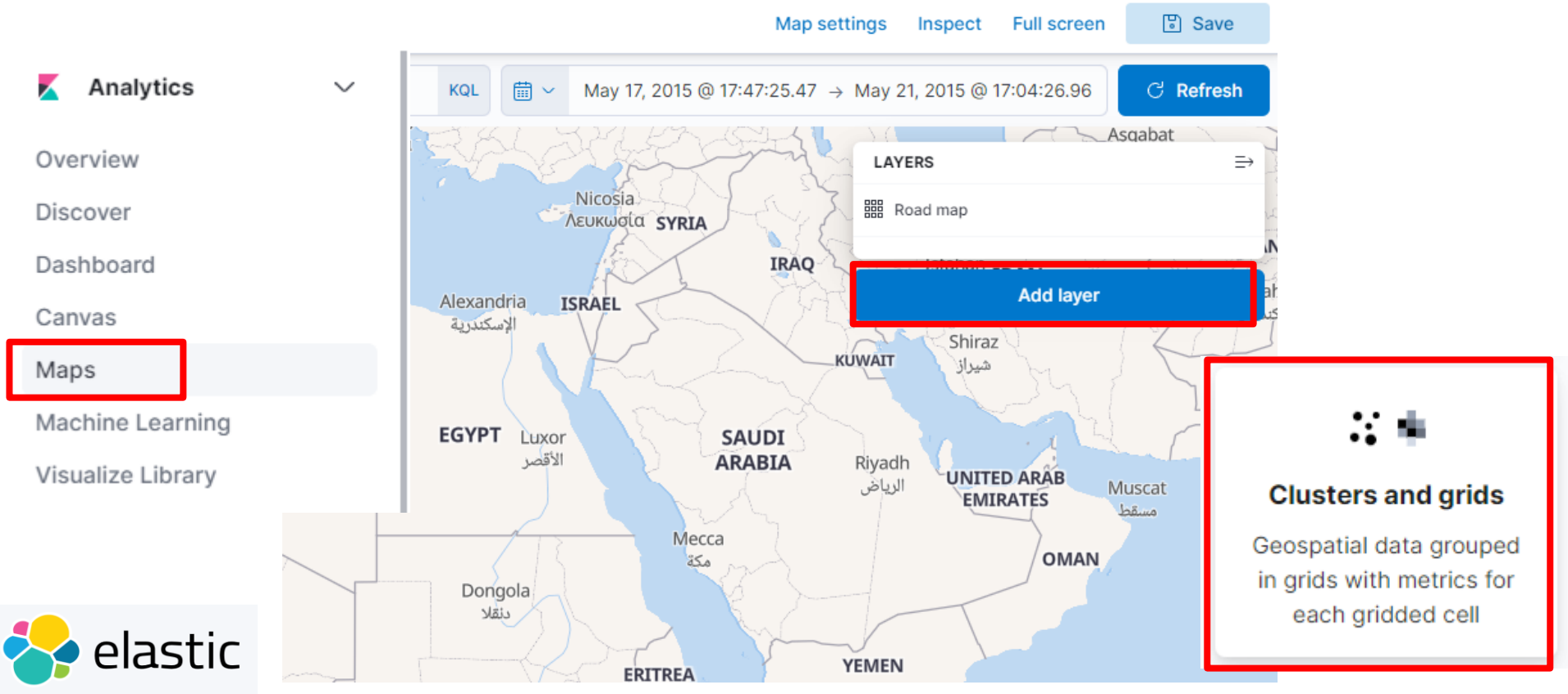
Kibana 시각화: Visualize

- shakespear 관련 막대 그래프 그리기!
  - 인덱스를 shakespear로 변경하고 Bar vertical을 선택한 후 드래그엔 드롭으로 적정한 데이터를 x,y에 지정
  - Bar Example로 저장



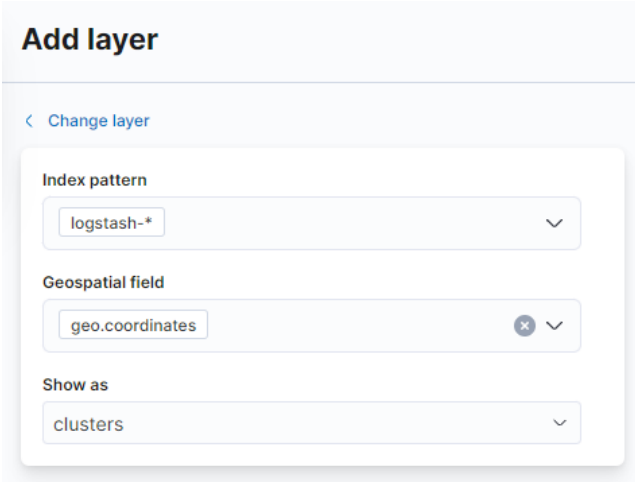
Kibana 시각화: Visualize

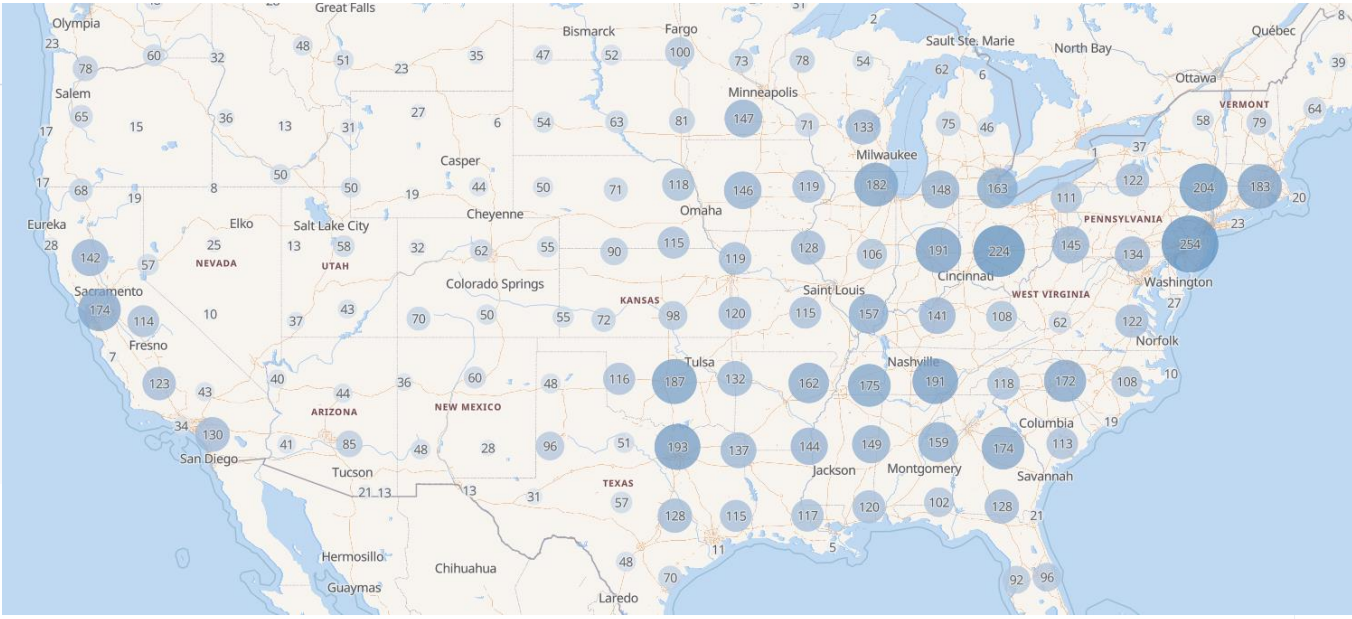
- Maps를 선택해서 logstash-*에 저장된 geo.coordinates 필드를 시각화
- Add Layer를 선택하고 clusters and grids를 선택



Kibana 시각화: Visualize

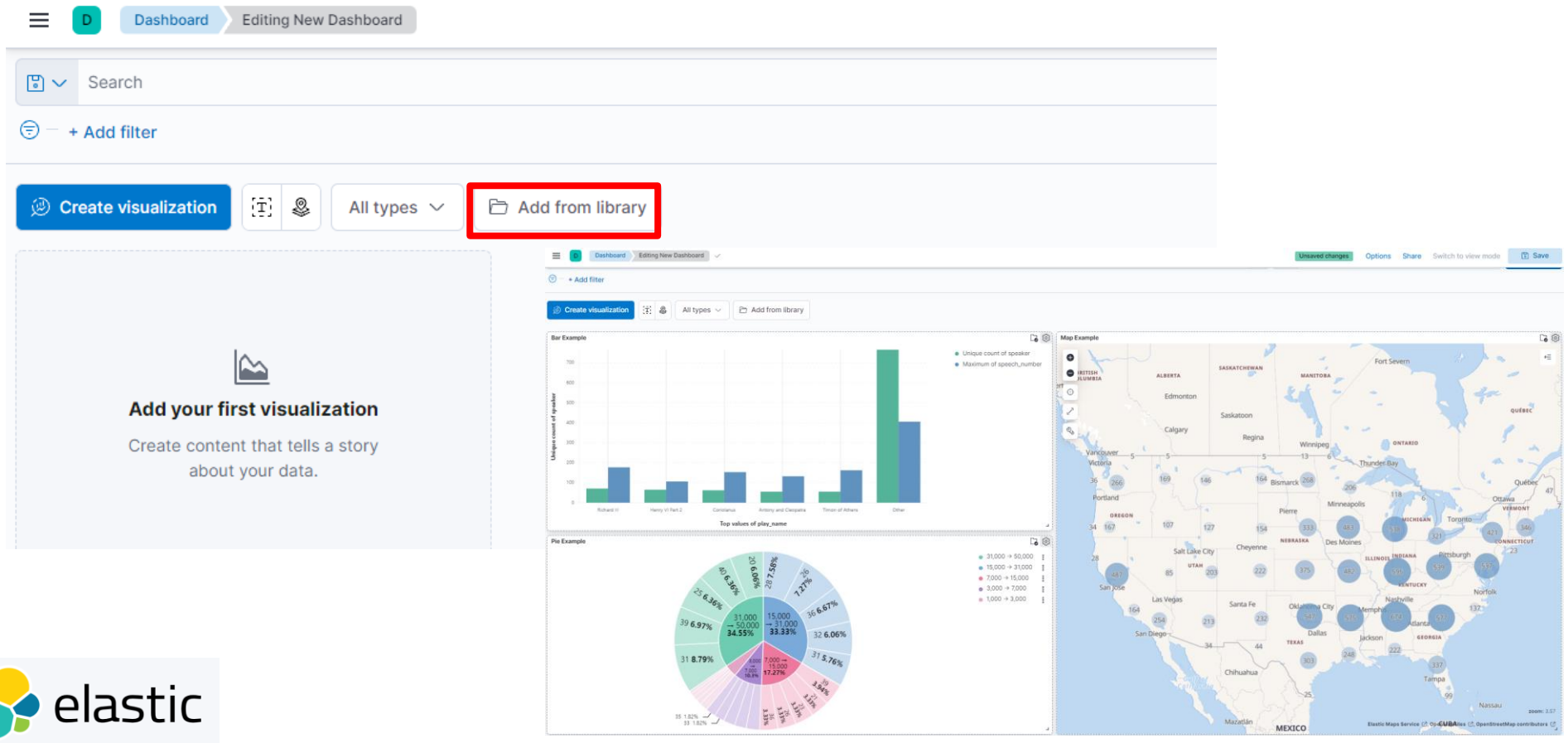
- logstash-*의 geo.coordinates 필드 클릭한 후 Add Layer를 선택
- Map Example로 저장





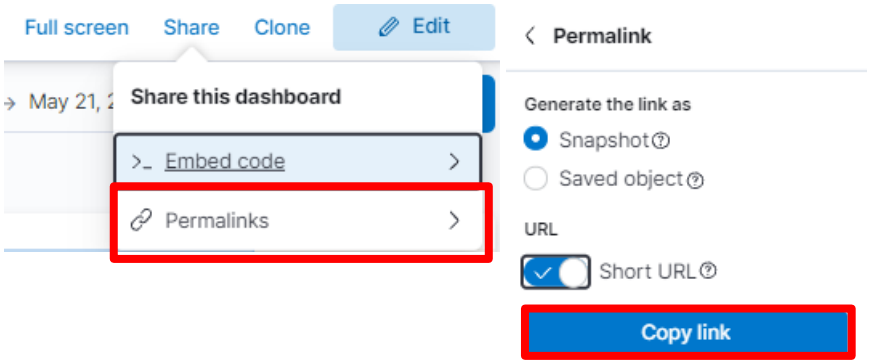
Kibana 시각화: Dashboard

- 대시보드 탭으로 이동하여 생성한 라이브러리를 가져오고 적절히 배치 후 Example로 저장

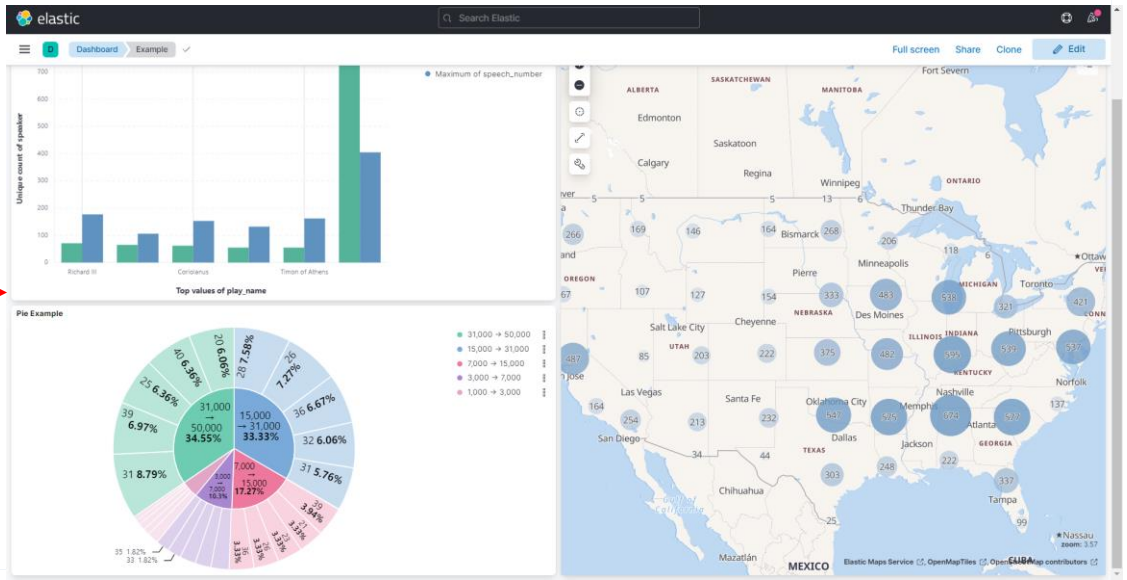


Kibana 시각화: Dashboard

- 대시보드 탭으로 이동하여 생성한 라이브러리를 가져오고 적절히 배치 후 Example로 저장

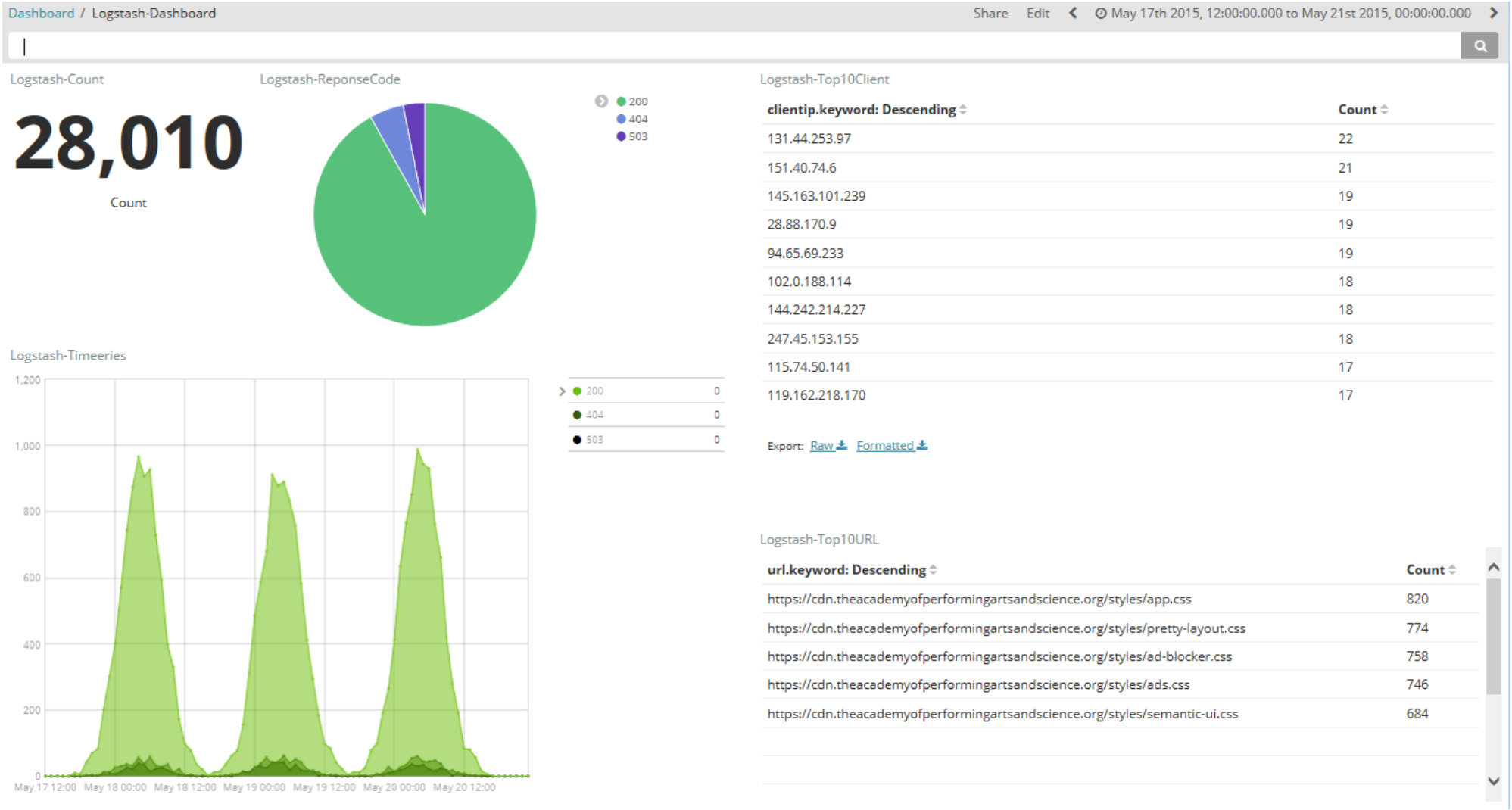


외부로 바로 공유 가능

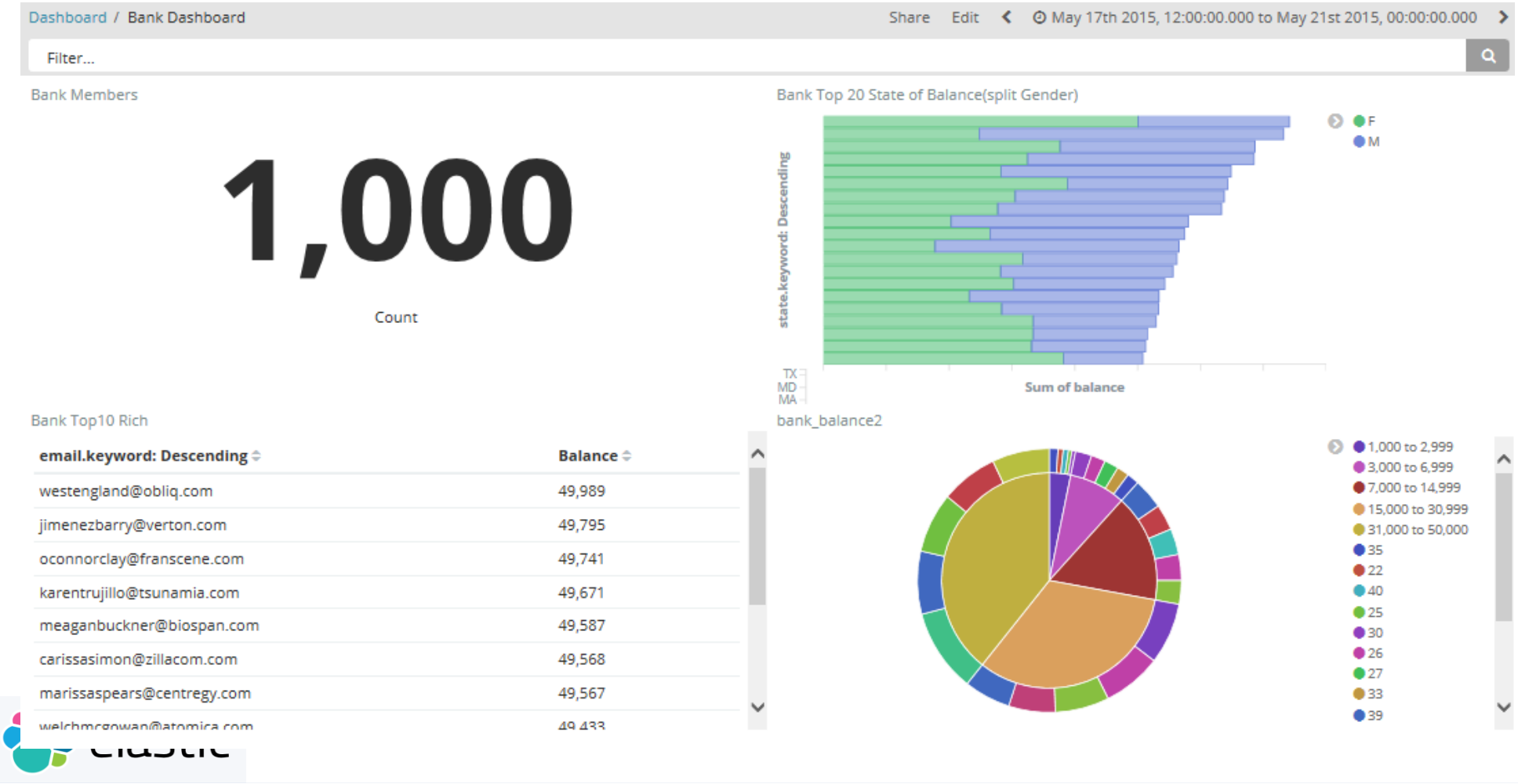


KIBANA 시각화

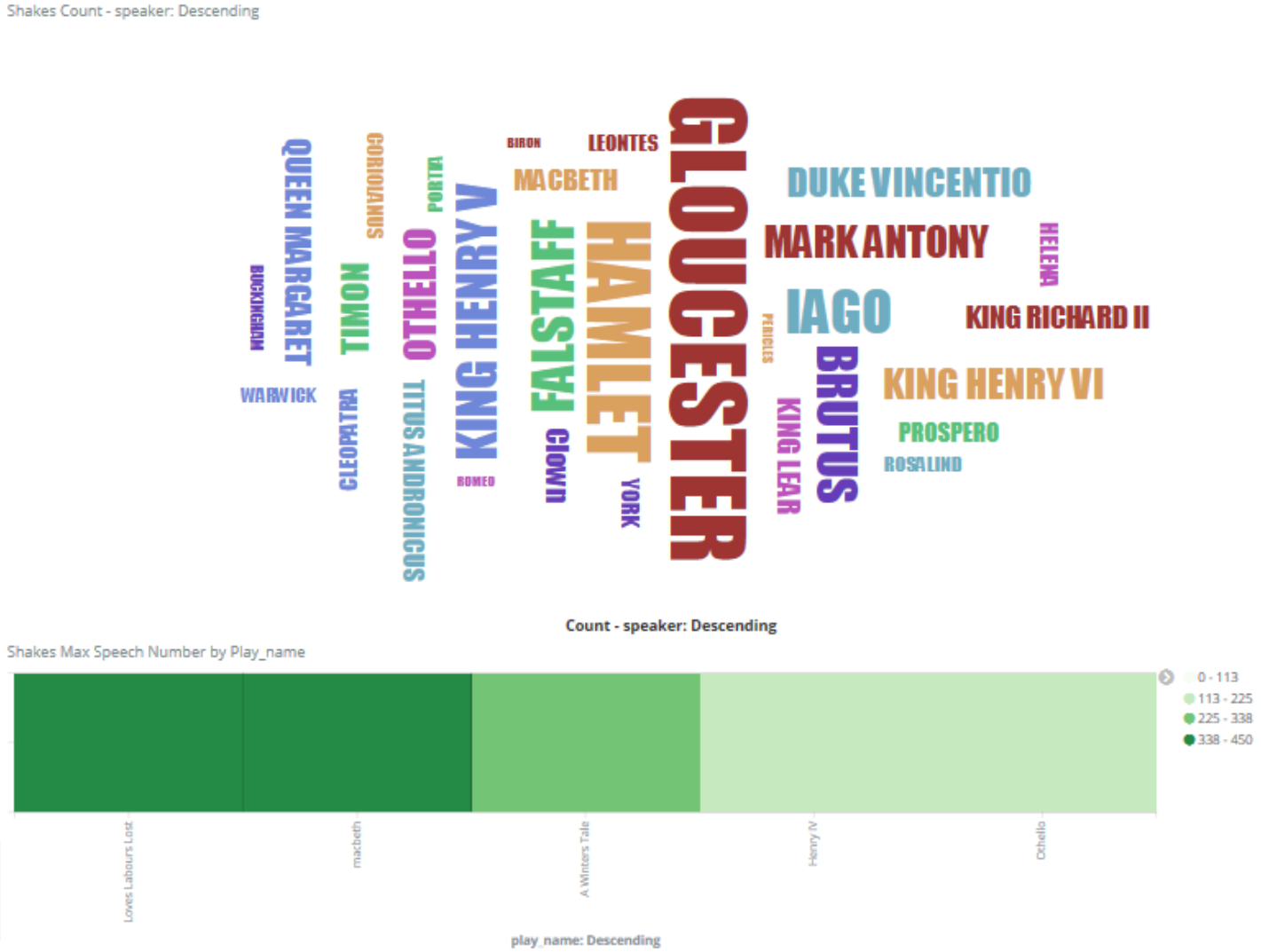
Kibana 시각화: 연습문제 Logstash



Kibana 시각화: 연습문제 Bank



Kibana 시각화: 연습문제 Bank

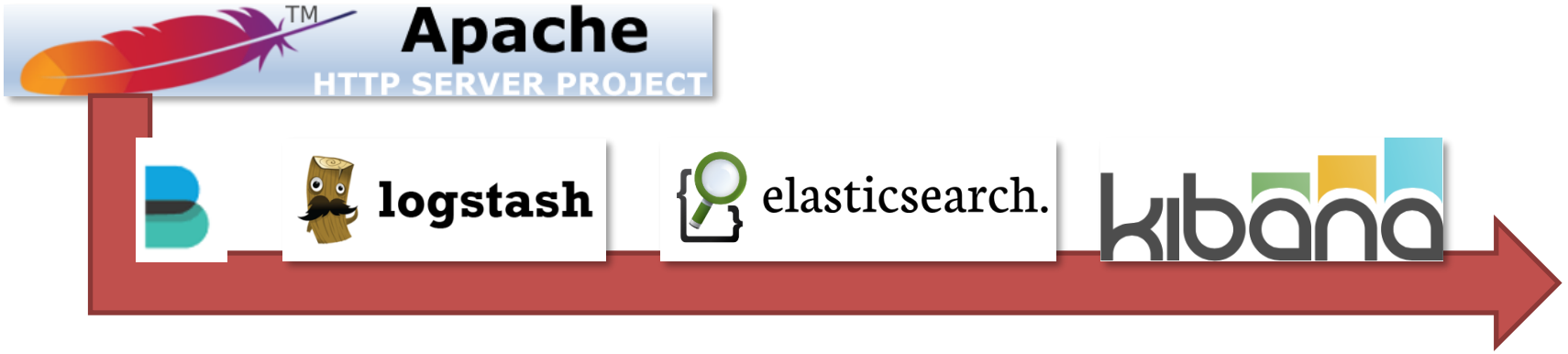


파일 비트를 활용한 아파치 서버 로그 수집

파일 비트를 활용한 아파치 서버 로그 수집

LogStash/Filebeat webservice 로그 자동 수집

- Ubuntu환경의 아파치2 설치 후 액세스 로그 저장
- Filebeat의 기능을 사용하여 파일을 긁어옴
- Logstash는 apache 파싱 기능을 사용해 데이터의 필드를 각각 분할하여 엘라스틱 서치에 전송



파일 비트를 활용한 아파치 서버 로그 수집

파일비트와 apache2 설치

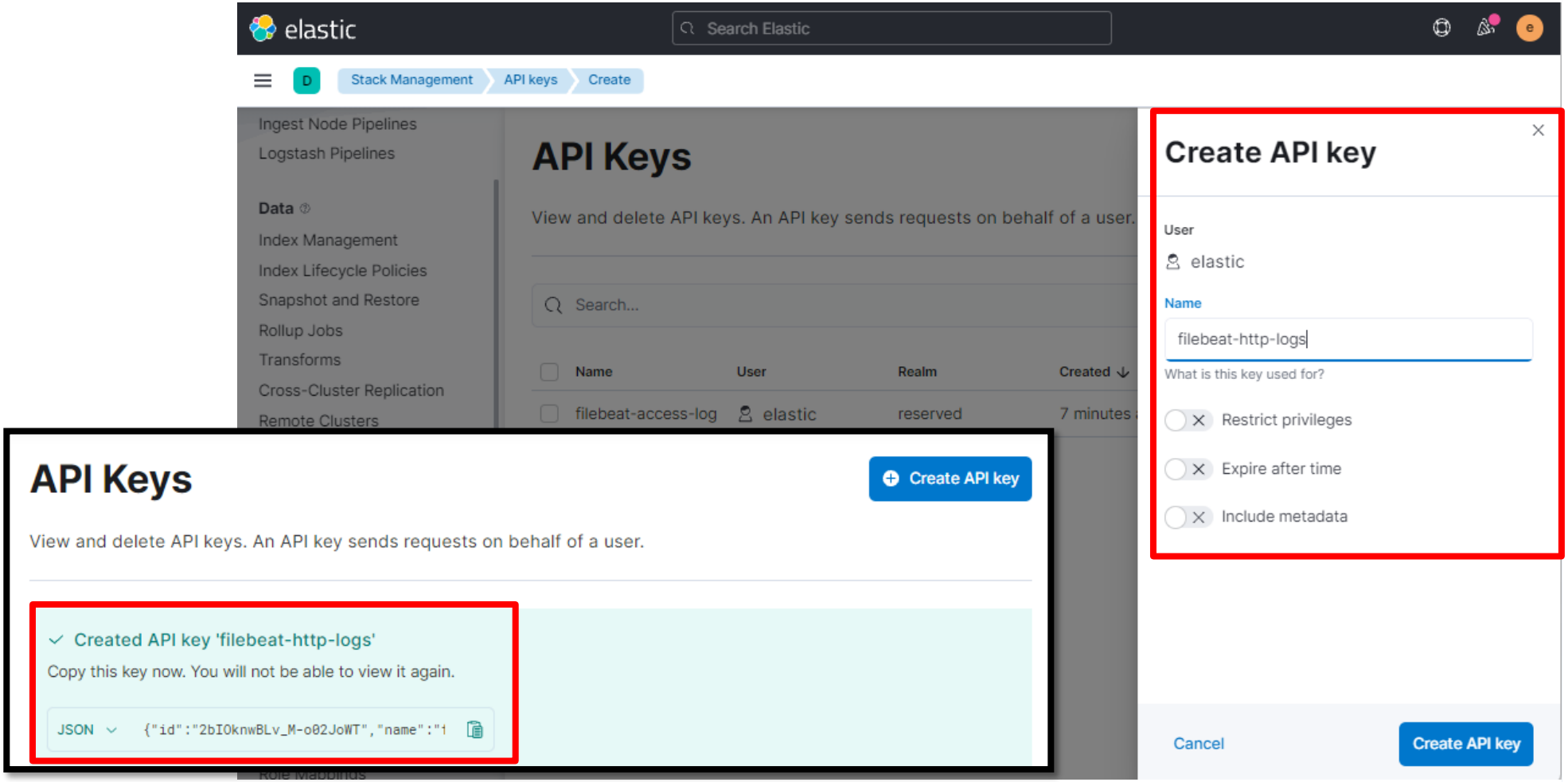
- **아파치 설치하기**
  - apt install apache2 -y

- **파일비트 다운로드 및 설치**
  - 파일비트 7.15.1 deb 32비트 다운로드 명령어 실행
  - wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.15.1-amd64.deb --no-check-certificate
  - sudo dpkg -i filebeat-7.15.1-amd64.deb

파일 비트를 활용한 아파치 서버 로그 수집

API 키 발급하기

- Stack Management - API keys - Create에서 API 키 생성



파일 비트를 활용한 아파치 서버 로그 수집

엘라스틱서치 접속 정보 입력

- /etc/filebeat/filebeat.yml 파일 수정
  - sudo vim /etc/filebeat/filebeat.yml
  - output.elasticsearch를 찾아서 호스트 위치가 정확한지 확인
  - https 프로토콜을 사용하도록 체크
  - 앞서 생성한 id와 apikey를 입력
  - 안전하지 않은 SSL 통신을 위해 mode를 none으로 설정

```
# ------------------------------ Elasticsearch Output ------------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["localhost:9200"]

  # Protocol - either `http` (default) or `https`.
  protocol: "https"

  # Authentication credentials - either API key or username/password.
  api_key: "yrIGknwBLv_M-o02UYUC:t8SlUwl2RCewwRHRXsi3wg"
  ssl.verification_mode: none
```

파일 비트를 활용한 아파치 서버 로그 수집

파일비트 모듈 설정으로 apache2 로그 수집

- **모듈 디렉토리에 apache.yml을 활성화하고 필요한 데이터를 입력**
  - cd /etc/filebeat/modules.d
  - cp apache.yml.disabled apache.yml
  - vim apache.yml

```
- module: apache
  access:
    enabled: true
    var.paths: ["/var/log/apache2/access.log*"]
  error:
    enabled: true
    var.paths: ["/var/log/apache2/error.log*"]
```

- **설정 파일 적용을 위해 재시작**

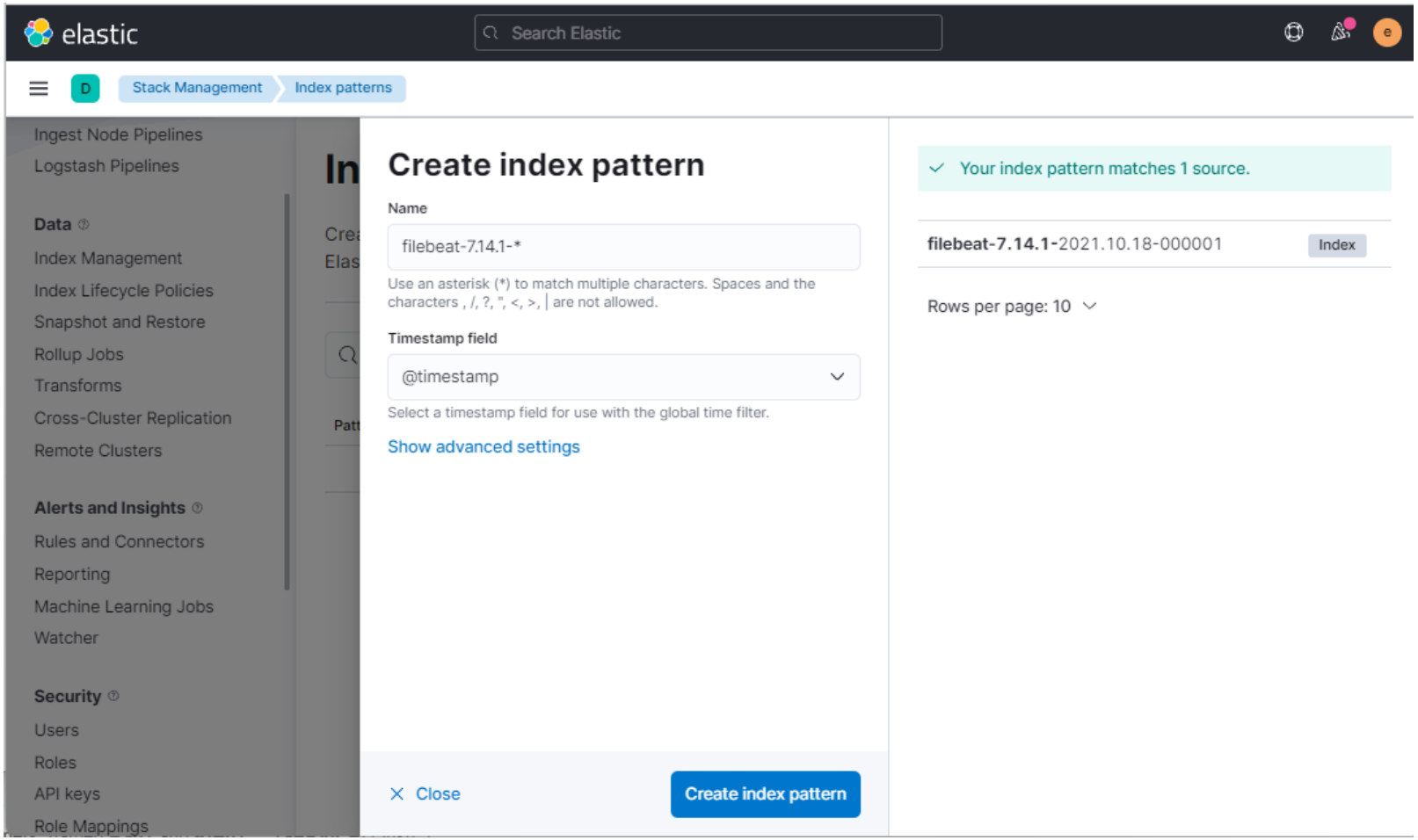
```
sudo /etc/init.d/filebeat restart
journalctl -u filebeat # 로그 확인
```

  - 오류 발생 시 확인 및 조치

파일 비트를 활용한 아파치 서버 로그 수집

대시보드에 파일비트

- 인덱스 패턴 확인 및 추가



파일 비트를 활용한 아파치 서버 로그 수집

디스커버에서 정보 확인

- Suricata 필드에서 파싱된 로그 확인