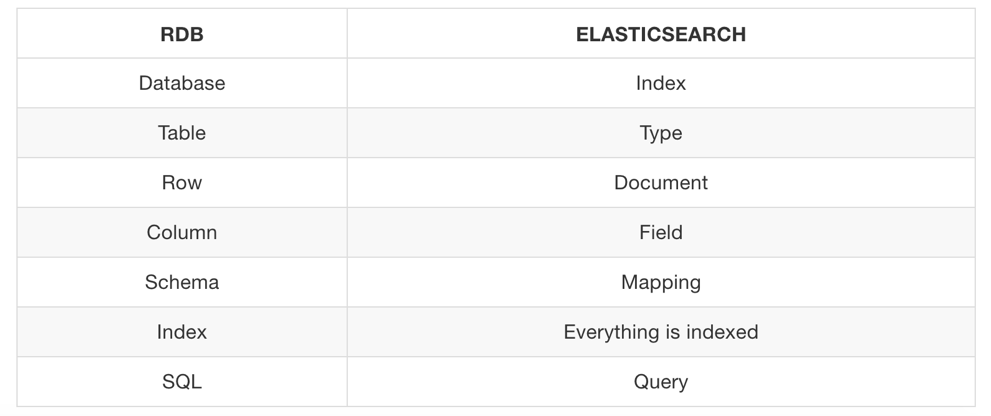
**Elastic Search**



시작-끄기

sudo service elasticsearch start

sudo service elasticsearch stop

<https://sanghaklee.gitbooks.io/elk/content/>

데이터 읽을 때(조회):

curl -XGET localhost:9200/classes 인덱스 이름!

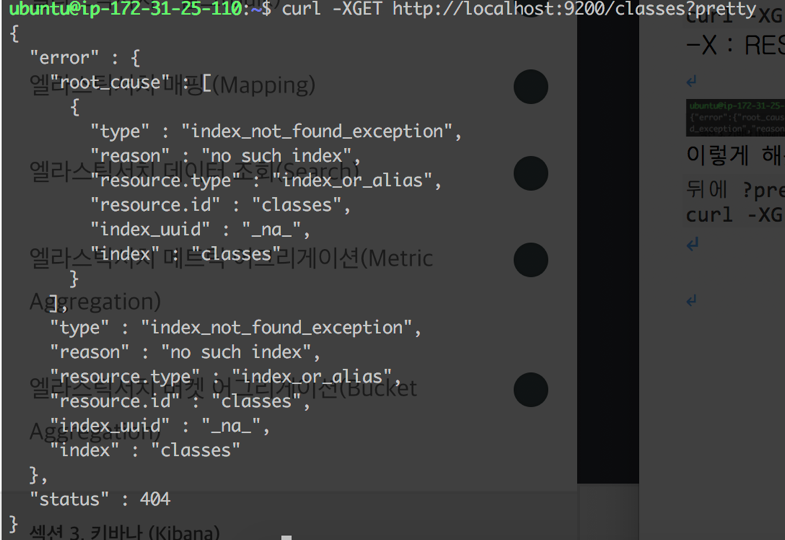
-X : REST API를 보낼 때prefix



이렇게 해봤는데 status가 404면 그 Index가 존재하지 않는다는 것!

뒤에 ?pretty를 붙이면 예쁘게 json 형식으로 데이터가 나온다

curl -XGET localhost:9200/classes?pretty

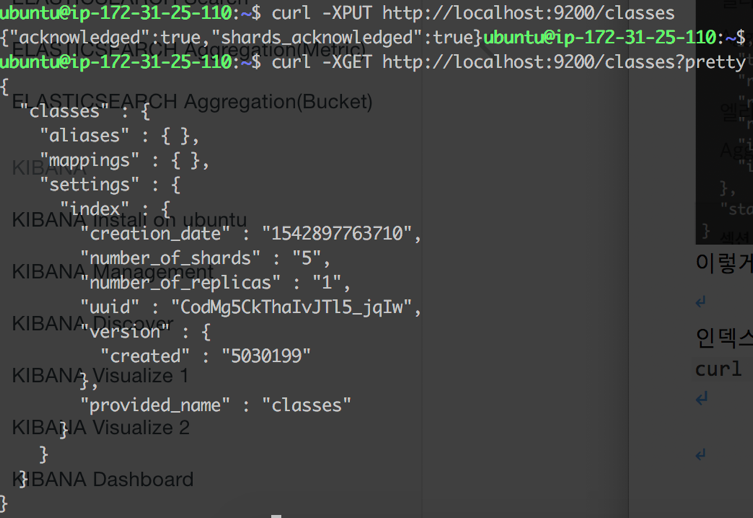


이렇게 예쁘게~

인덱스 생성

curl -XPUT localhost:9200/classes

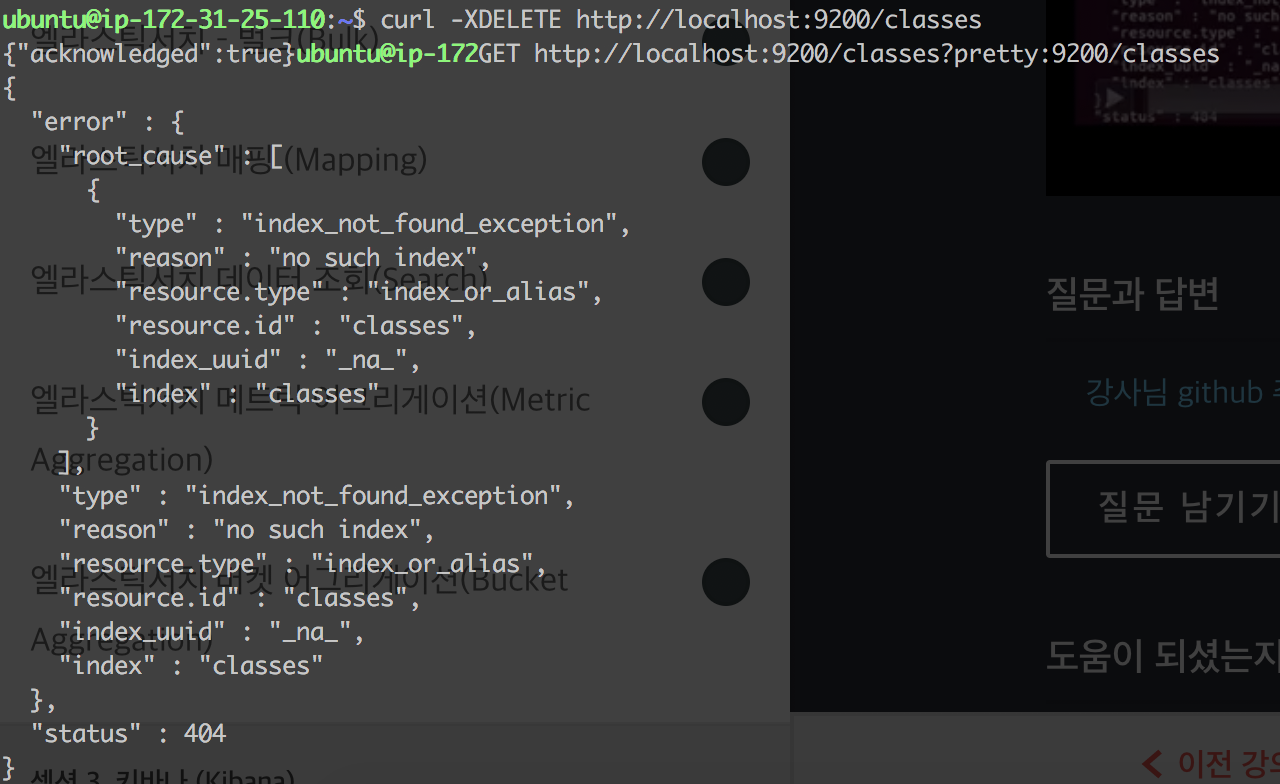
생성하고 다시 GE으로 확인해보면 이렇게 인덱스가 잘 생성된 모습 확인!



인덱스 삭제

curl -XDELETE localhost:9200/classes

삭제 하고 확인해보면 acknowledged = true 라고 확인해줌



**Document 생성**

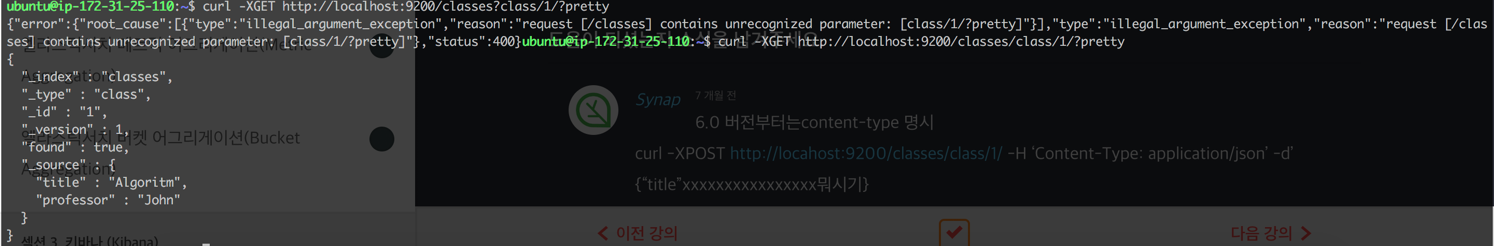
인덱스가 없더라도, 인덱스명, doc명, type명을 명시하면 바로 doc 생성 가능

curl -XPOST localhost:9200/classes/class/1/ -d '{"title":"A", "professor":"J"}'

주소 / 인덱스명 / 타입명 / 아이디 -d ‘json내용- 타입:값 ’

참고) 최근 버전인 6.xx버전은 post시에 content-type을 입력해야 하네요. 5.xx버전으로 설치하고 가상머신에서 작동시에는 메모리가 2G이상이여야 작동합니다..

다시 GET으로 확인해보면 이렇게나 잘 나온다.



또는, 파일에서 doc생성하기 - @파일이름



이렇게나 잘 되는 모습. 성공했을 때는 successful에 갯수가 나오고, status가 안나온다는 특징이 있다

다시 GET으로 확인해본다.

**UPDATE**

**기존에 있는 Document에 하나의 Field 추가하기**

curl -XPOST http://localhost:9200/classes/class/1/\_update -d '{"doc":{"unit":1}}'



이렇게나 잘 새로운 Field가 들어간 모습!

**기존에 있는 Document의, 기존의 Field 값 수정하기**

그냥 아까랑 똑같이 !!! 바꾸고 싶은 값만 바꿔서 넣으면 된다~

curl -XPOST http://localhost:9200/classes/class/1/\_update -d '{"doc":{"unit":2}}'

* **참고 : 중간에 pretty 옵션 주기**

curl -XPOST http://localhost:9200/classes/class/1/\_update?pretty -d '{"doc":{"unit":2}}'

**방법2. Script이용 : 좀 프로그래밍스럽게 할 수 있음**

curl -XPOST localhost:9200/classes/class/1/\_update -d '{"script":"ctx.\_source.unit +=5"}'

**여러개의 Document를 한번에 삽입하는 방법 – Bulk(벌크)**

-bulk옵션을 주면 여러개의 document를 한번에 삽입할 수 있다.

curl -XPOST http://localhost:9200/\_bulk?pretty --data-binary @classes.json

벌크는 파일로!

2개의 라인으로 구성

첫번째 라인 : meta information, 어떤 인덱스, 어떤 타입, 어떤 아이디

두번째 라인: document 내용

Classes.json 내용)

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "1" } }

{"title" : "Machine Learning","Professor" : "Minsuk Heo","major" : "Computer Science","semester" : ["spring", "fall"],"student\_count" : 100,"unit" : 3,"rating" : 5, "submit\_date" : "2016-01-02", "school\_location" : {"lat" : 36.00, "lon" : -120.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "2" } }

{"title" : "Network","Professor" : "Minsuk Heo","major" : "Computer Science","semester" : ["fall"],"student\_count" : 50,"unit" : 3,"rating" : 4, "submit\_date" : "2016-02-02", "school\_location" : {"lat" : 36.00, "lon" : -120.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "3" } }

{"title" : "Operating System","Professor" : "Minsuk Heo","major" : "Computer Science","semester" : ["spring"],"student\_count" : 50,"unit" : 3,"rating" : 4, "submit\_date" : "2016-03-02", "school\_location" : {"lat" : 36.00, "lon" : -120.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "5" } }

{"title" : "Machine Learning","Professor" : "Tim Cook","major" : "Computer Science","semester" : ["spring"],"student\_count" : 40,"unit" : 3,"rating" : 2, "submit\_date" : "2016-04-02", "school\_location" : {"lat" : 39.00, "lon" : -112.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "6" } }

{"title" : "Network","Professor" : "Tim Cook","major" : "Computer Science","semester" : ["summer"],"student\_count" : 30,"unit" : 3,"rating" : 2, "submit\_date" : "2016-02-02", "school\_location" : {"lat" : 36.00, "lon" : -120.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "7" } }

{"title" : "Operating System","Professor" : "Jeniffer Anderson","major" : "Computer Science","semester" : ["winter"],"student\_count" : 30,"unit" : 3,"rating" : 1, "submit\_date" : "2016-11-02", "school\_location" : {"lat" : 39.97, "lon" : -89.78}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "8" } }

{"title" : "Algorithm","Professor" : "Tim Cook","major" : "Computer Science","semester" : ["fall"],"student\_count" : 80,"unit" : 3,"rating" : 2, "submit\_date" : "2016-10-22", "school\_location" : {"lat" : 39.97, "lon" : -89.78}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "9" } }

{"title" : "Data Structure","Professor" : "Tim Cook","major" : "Computer Science","semester" : ["winter"],"student\_count" : 50,"unit" : 3,"rating" : 2, "submit\_date" : "2016-07-22", "school\_location" : {"lat" : 39.97, "lon" : -89.78}}

삽입 하고나서 하나하나 조회해보면 잘 들어가 있다.

curl -XGET <http://localhost:9200/classes/class/2?pretty>

**엘라스틱서치 매핑(Mapping)**

* Mapping은 RDBMS에서 스키마(Schema)와 동일한 개념!
* 효율적인 검색을 위해서 **데이터의 타입**을 정의하는 것

매핑 없이 엘라스틱 서치를 다룰 수 있을까? YES.

하지만, 매핑을 안쓰는건 실제 업무에서 굉장히 위험한 일이다.

날짜, 숫자를 넣을 때 엘라스틱 서치가 문자로 넣을 수도 있다.

잘못 지정된 타입은 키바나로 시각화할때 값 처리가 마음대로 잘 되지 않을 수 있다.

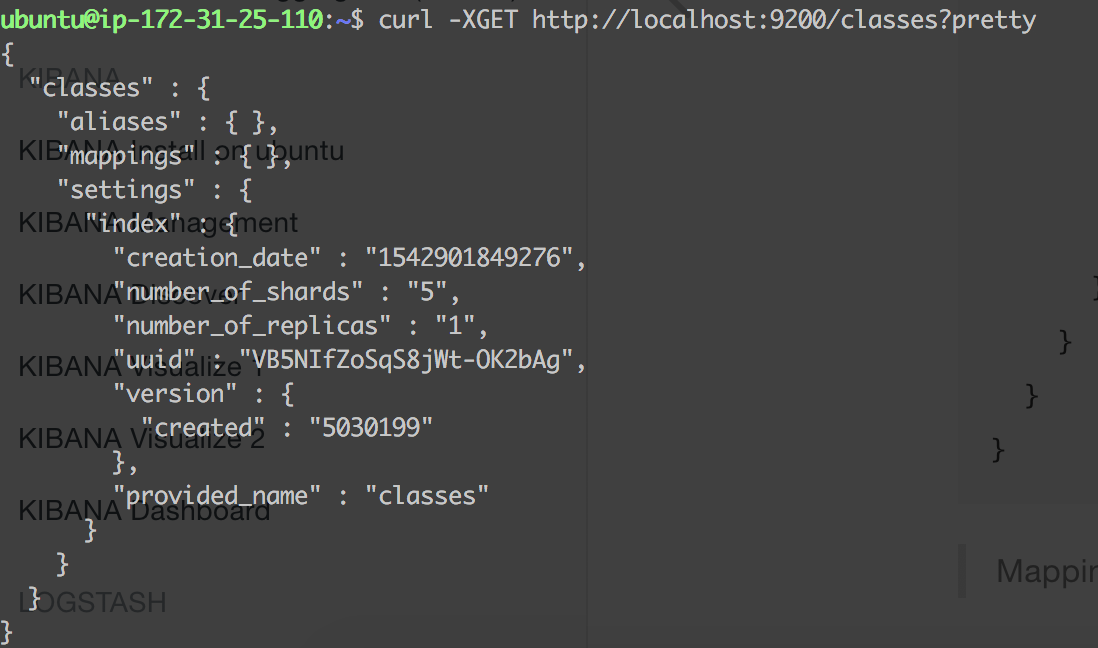
**인덱스 생성하기**

curl -XPUT http://localhost:9200/classes

**확인하기**

curl -XGET http://localhost:9200/classes

주의깊게 보면 매핑이 없다



**매핑 추가하기 – PUT 과 \_mapping**

미리 만들어둔 mapping파일로 추가한다.

curl -XPUT http://localhost:9200/classes/class/\_mapping -d @classesRating\_mapping.json

파일 내용) classesRating\_mapping.json

{

"class" : {

"properties" : {

"title" : {

"type" : "string"

},

"professor" : {

"type" : "string"

},

"major" : {

"type" : "string"

},

"semester" : {

"type" : "string"

},

"student\_count" : {

"type" : "integer"

},

"unit" : {

"type" : "integer"

},

"rating" : {

"type" : "integer"

},

"submit\_date" : {

"type" : "date",

"format" : "yyyy-MM-dd"

},

"school\_location" : {

"type" : "geo\_point"

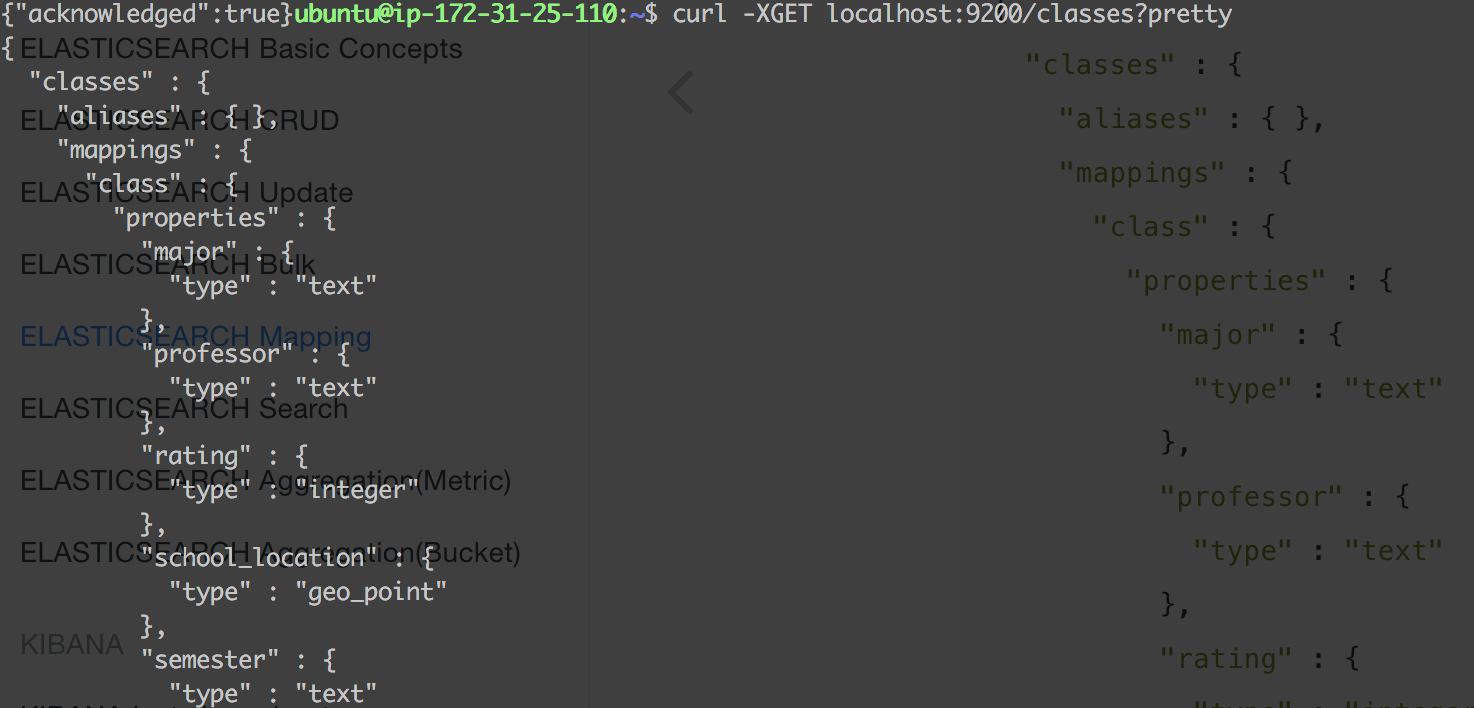
}

}

}

**다시 확인**

curl -XGET http://localhost:9200/classes?pretty



아까는 mapping이 비어있었는데 지금은 이렇게나 잘 들어간다.

**실제 데이터 넣기 – bulk옵션으로 한번에**

curl -XPOST localhost:9200/\_bulk?pretty --data-binary @classes.json

**다시 확인하기**

curl -XGET localhost:9200/classes/class/1?pretty

너무나 잘 들어간 모습 확인 가능! (데이터 타입 맞춰서 들어가있다)

엘라스틱 서치 매핑(Mapping)

매핑은 RDBMS에서 스키마(Schema)와 동일한 개념 !

참고사항)

6.1버전 기준으로 string 타입은 에러나네요. text 사용하시면 됩니다.

6.x버전 이후 curl 다음에 -H ‘Content-Type: application/json’을 넣어줘야 합니다.

string 타입 때문에 에러나는 경우 unix 계열에서는 다음과같은 명령어로 string 을 text 로 변환하세요.

sed “s/string/text/” classesRating\_mapping.json | tee classesRating\_mapping.json

**Search**

* 데이터 조회!

**먼저 bulk로 document 삽입**

curl -XPOST localhost:9200/\_bulk --data-binary @simple\_basketball.json

2개의 document가 담긴 파일, 파일 내용) simple\_basket.json

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "1" } }

{"team" : "Chicago Bulls","name" : "Michael Jordan", "points" : 30,"rebounds" : 3,"assists" : 4, "submit\_date" : "1996-10-11"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "2" } }

{"team" : "Chicago Bulls","name" : "Michael Jordan","points" : 20,"rebounds" : 5,"assists" : 8, "submit\_date" : "1996-10-11"}

**Search 기본**

curl -XGET localhost:9200/basketball/record/\_search?pretty

인덱스이름 / 타입이름\_serach키워드

2개의 document가 확인된다!

**Search 자세한 검색 => 추후 공부 필요**

**방법 1. URI : 쿼리문(?) 넣기!**

* 조건에 해당하는 document만 조회된다.
* Pretty옵션도 같이 주려면, 전체를 ‘’로 감싸야 한다……ㅠㅠ

curl -XGET ‘http://localhost:9200/basketball/record/\_search?q=points:30&pretty’

**방법2. Request Body : Request의 body를 사용하여 넣기**

* 무슨 query인지 언급(무슨 조건인지, 여기서는 term에 대한 조건이므로 term이라고 언급)
* Request Body 방법은 다양한 option이 존재하는데, 공식 doc을 참고하자

<https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-body.html>

curl -XGET http://localhost:9200/basketball/record/\_search?pretty -d '{

"query": {

"term": {

"points": 30

}

}

}'

**Aggregation(Metric)**

* 엘라스틱서치안의 어떤 document에서 조합을 통해서 어떤 값을 도출해내는 방법
* 그 중, Metric은 산술 할 때 쓰인다. (산술분석)

예) 평균, 최댓값, 최솟값

**예제1) 평균 구하기**

curl -XGET http://localhost:9200/\_search?pretty --data-binary @avg\_points\_aggs.json

Aggregation 파일 내용- avg\_points\_aggs.json )

{

"size" : 0,

"aggs" : {

"avg\_score" : {

"avg" : { 사용할 aggs 종류

"field" : "points" 어떤 평균을 구할거냐?

}

}

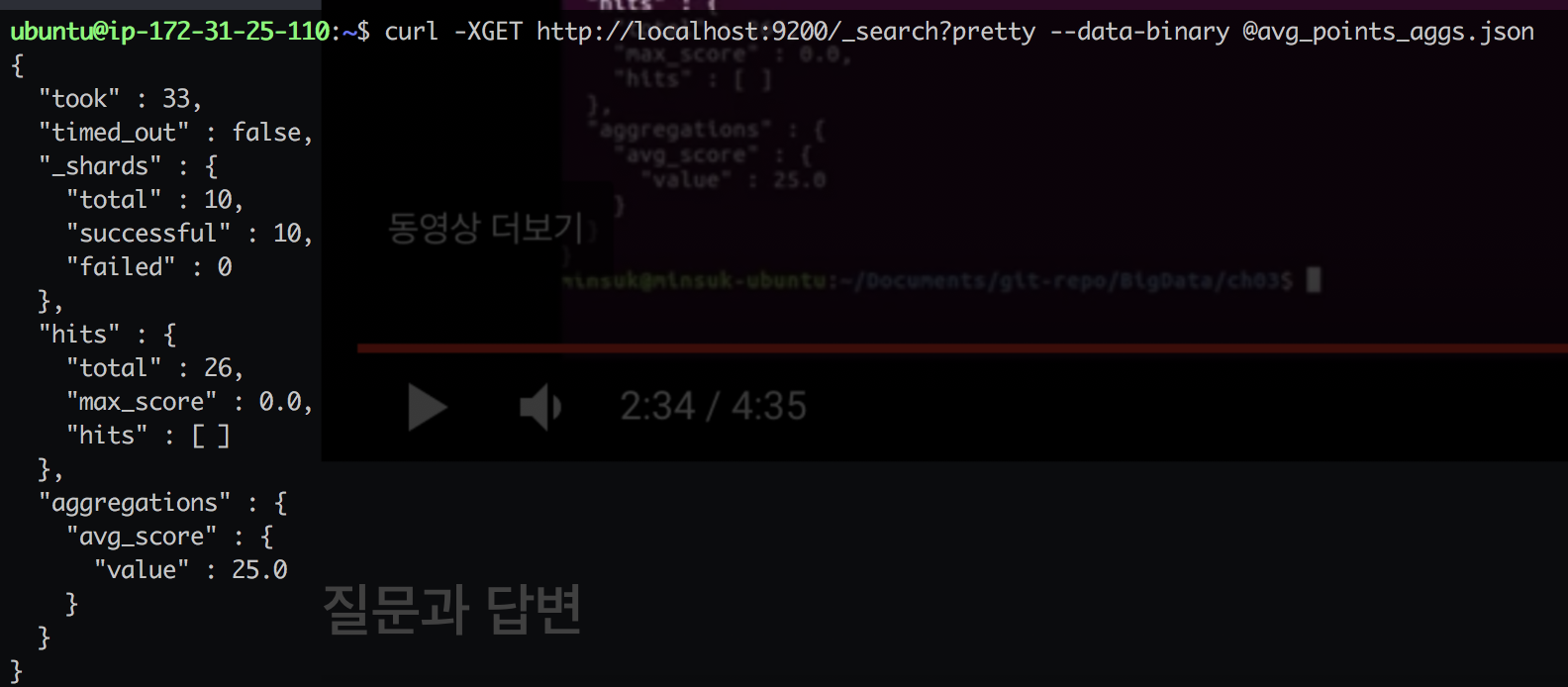
}

}

Size: 우리가 보고싶은 것만 결과를 보기 위해

Aggs : aggregations

아래와 같이 실행시켜보면, aggregations의 value에 값이 아주 잘 나온다.



예제2) 최대값 구하기 - max

curl -XGET http://localhost:9200/\_search?pretty --data-binary @max\_points\_aggs.json

Aggregation 파일 내용- max\_points\_aggs.json

{

"size": 0,

"aggs": {

"max\_score": {

"max": {

"field": "points"

}

}

}

}

예제3) 최솟값 구하기 - min

curl -XGET http://localhost:9200/\_search?pretty --data-binary @min\_points\_aggs.json

파일 내용

{

"size": 0,

"aggs": {

"min\_score": {

"min": {

"field": "points"

}

}

}

}

예제 4) 합 구하기 – sum

curl -XGET http://localhost:9200/\_search?pretty --data-binary @sum\_points\_aggs.json

\*\* 전체 Document에서 field가 “points”인 것들의 sum을 구해주세요!! \*\*

파일 내용

{

"size": 0,

"aggs": {

"sum\_score": {

"sum": {

"field": "points"

}

}

}

}

**평균, 합, 최소, 최대 한 번에 도출!! - Stats**

curl -XGET http://localhost:9200/\_search?pretty --data-binary @stats\_points\_aggs.json

파일 내용)

{

"size": 0,

"aggs": {

"stats\_score": {

"stats": {

"field": "points"

}

}

}

}

**Aggregation(Bucket)**

* Group by라고 보면 된다. 데이터를 일정 기준으로 묶어서 결과를 도출.
* Document를 팀별로 그룹지어서!!

**예제)**

**인덱스 생성하기**

curl -XPUT http://localhost:9200/basketball

**매핑 넣기**

curl -XPUT http://localhost:9200/basketball/record/\_mapping -d @basketball\_mapping.json

내용) record라는 Type안에 각각의 property의 type을 정의한다

fileddata: true를 해서, term aggregation 할 때 이 필드로 조회할 수 있게 함

파일 내용)

{

"record" : {

"properties" : {

"team" : {

"type" : "string",

"fielddata" : true

},

"name" : {

"type" : "string",

"fielddata" : true

},

"points" : {

"type" : "long"

},

"rebounds" : {

"type" : "long"

},

"assists" : {

"type" : "long"

},

"blocks" : {

"type" : "long"

},

"submit\_date" : {

"type" : "date",

"format" : "yyyy-MM-dd"

}

}

}

}

**Document 삽입하기(\_bulk)**

curl -XPOST http://localhost:9200/\_bulk --data-binary @twoteam\_basketball.json

파일 내용)

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "1" } }

{"team" : "Chicago","name" : "Michael Jordan", "points" : 30,"rebounds" : 3,"assists" : 4, "blocks" : 3, "submit\_date" : "1996-10-11"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "2" } }

{"team" : "Chicago","name" : "Michael Jordan","points" : 20,"rebounds" : 5,"assists" : 8, "blocks" : 4, "submit\_date" : "1996-10-13"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "3" } }

{"team" : "LA","name" : "Kobe Bryant","points" : 30,"rebounds" : 2,"assists" : 8, "blocks" : 5, "submit\_date" : "2014-10-13"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "4" } }

{"team" : "LA","name" : "Kobe Bryant","points" : 40,"rebounds" : 4,"assists" : 8, "blocks" : 6, "submit\_date" : "2014-11-13"}

**Term aggs**

* Group만들기..? 같은 개념
* 결과로 각각의 그룹에 해당하는 document가 몇 개 있는지 나온다.

curl -XGET http://localhost:9200/\_search?pretty --data-binary @terms\_aggs.json

파일 내용)

{

"size" : 0,

"aggs" : {

"players" : {

"terms" : {

"field" : "team" //team이라는 필드로 Term을 만들꺼야!

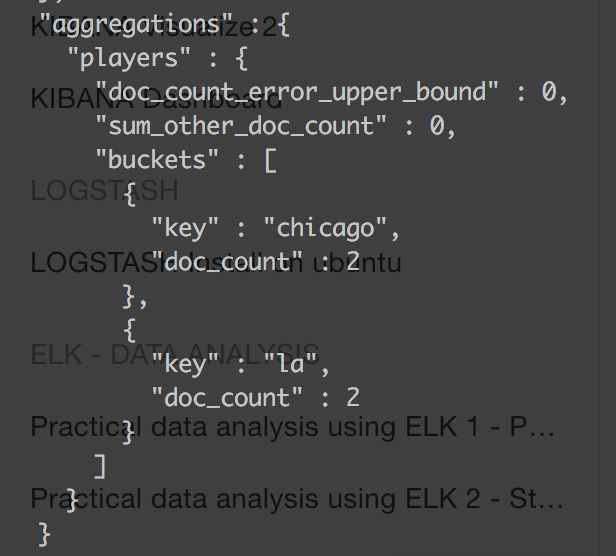
}

}

}

}

하면 결과가)



이렇게 buckets에 Term에 해당하는 doc이 몇개인지 알려준다

**Aggs(Stats group by team)**

curl -XGET http://localhost:9200/\_search?pretty --data-binary @stats\_by\_team.json

파일 내용)

일단 term을 사용해서 “team”별로 document를 묶는다.

그 결과값에서 다시 aggs를 넣는다. 이제 “team”별로 묶인 document들 중에서 stats라는 aggregation을 쓴다(filed는 points! 점수별로 stats를 보여줘라)

{

"size" : 0,

"aggs" : {

"team\_stats" : {

"terms" : {

"field" : "team"

},

"aggs" : {

"stats\_score" : {

"stats" : {

"field" : "points"

}

}

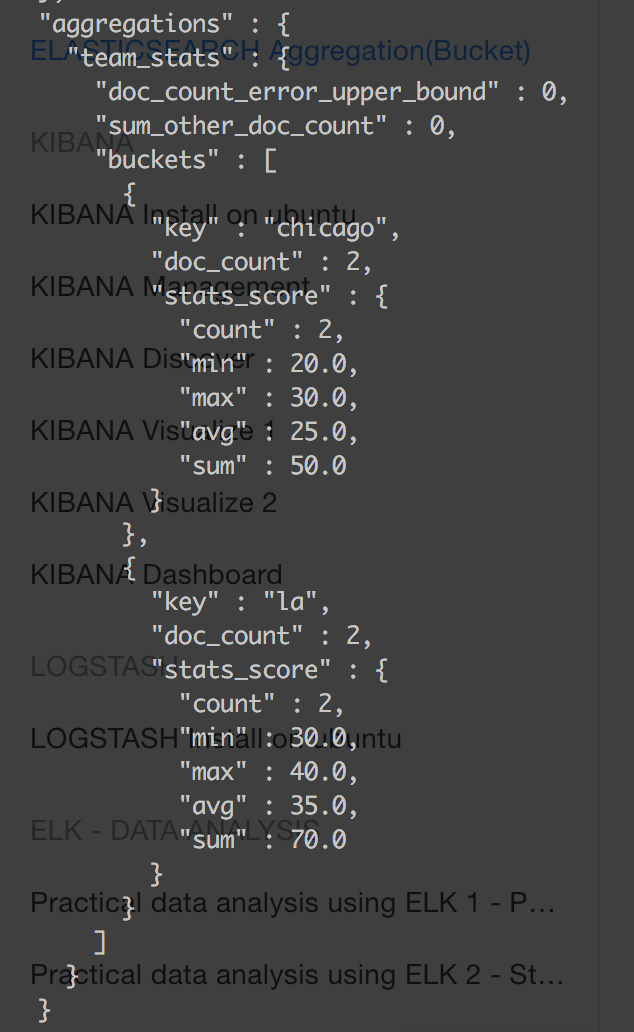
}

}

}

}

이렇게 내가 지정한 term 별로 결과가 나온다



**KIBANA**

**설치하기**

wget <https://artifacts.elastic.co/downloads/kibana/kibana-5.3.1-amd64.deb>

sudo dpkg -i kibana-5.3.1-amd64.deb

**Config바꿔주기**

vim /etc/kibana/kibana.yml

#server.host: "localhost"

#elasticsearch.url: "[http://localhost:9200](http://localhost:9200/)"

**키바나 시작하기**

sudo /usr/share/kibana/bin/kibana

**브라우저에서 열기**

나는 클라우드 인스턴스를 사용하기 때문에, 내 로컬 브라우저에서 열려면, Kibana의 config를 조금 바꿔줘야 한다.

sudo vi /etc/kibana/kibana.yml

server.host = "0.0.0.0"

브라우저에서

Public Ip주소:5601로 접속한다

꼭!!Elastic search와 kibana를 모두 0.0.0.0으로 바꿔서 외부 접속 허용하기!!!!

해결방법)

Elasticsearch 외부 접속 확인하기

설정을 0.0.0.0으로 바꾼 후, 재시작

sudo service elasticsearch restart

아아 미쳤다 config 바꾸고 주석 안풀었다 ㅠㅠㅠㅠ

된다 된다된다ㅏ다다다다다다

**Kibana management**

우선, Elastic Search에 document 넣기

**인덱스 생성 후 매핑 넣어주기**

curl -XPUT localhost:9200/basketball/record/\_mappin -d @basketball\_mapping.json

파일 내용)

basketball\_mapping.json

{

"record" : {

"properties" : {

"team" : {

"type" : "string",

"fielddata" : true

},

"name" : {

"type" : "string",

"fielddata" : true

},

"points" : {

"type" : "long"

},

"rebounds" : {

"type" : "long"

},

"assists" : {

"type" : "long"

},

"blocks" : {

"type" : "long"

},

"submit\_date" : {

"type" : "date",

"format" : "yyyy-MM-dd"

}

}

}

}

**Bulk로 Document 추가**

curl -XPOST localhost:9200/\_bulk --data-binary @bulk\_basketball.json

파일 내용)

Bulk\_basketball.json

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "1" } }

{"team" : "Golden States Warriors","name" : "Stephen Curry", "points" : 30,"rebounds" : 3,"assists" : 4, "blocks" : 5, "submit\_date" : "2016-10-11"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "2" } }

{"team" : "Golden States Warriors","name" : "Stephen Curry","points" : 32,"rebound" : 5,"assist" : 8, "blocks" : 5, "submit\_date" : "2016-10-13"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "3" } }

{"team" : "Golden States Warriors","name" : "Stephen Curry","points" : 28,"rebound" : 2,"assist" : 3, "blocks" : 1, "submit\_date" : "2016-10-17"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "4" } }

{"team" : "Golden States Warriors","name" : "Stephen Curry","points" : 36,"rebound" : 1,"assist" : 2, "blocks" : 1, "submit\_date" : "2016-11-20"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "5" } }

{"team" : "Golden States Warriors","name" : "Stephen Curry","points" : 36,"rebound" : 1,"assist" : 2, "blocks" : 1, "submit\_date" : "2016-11-25"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "6" } }

{"team" : "Golden States Warriors","name" : "Stephen Curry","points" : 32,"rebound" : 1,"assist" : 4, "blocks" : 1, "submit\_date" : "2016-11-28"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "7" } }

{"team" : "Utah Jazz","name" : "Rudy Gobert", "points" : 3,"rebounds" : 11,"assists" : 4, "blocks" : 7, "submit\_date" : "2016-10-12"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "8" } }

{"team" : "Utah Jazz","name" : "Rudy Gobert","points" : 4,"rebound" : 13,"assist" : 8, "blocks" : 5, "submit\_date" : "2016-10-14"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "9" } }

{"team" : "Utah Jazz","name" : "Rudy Gobert","points" : 8,"rebound" : 10,"assist" : 3, "blocks" : 6, "submit\_date" : "2016-10-18"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "10" } }

{"team" : "Utah Jazz","name" : "Rudy Gobert","points" : 12,"rebound" : 9,"assist" : 2, "blocks" : 6, "submit\_date" : "2016-11-10"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "11" } }

{"team" : "Utah Jazz","name" : "Rudy Gobert","points" : 12,"rebound" : 14,"assist" : 2, "blocks" : 7, "submit\_date" : "2016-11-22"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "12" } }

{"team" : "Utah Jazz","name" : "Rudy Gobert","points" : 8,"rebound" : 10,"assist" : 4, "blocks" : 5, "submit\_date" : "2016-11-27"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "13" } }

{"team" : "Washington Wizards","name" : "John Wall","points" : 8,"rebound" : 1,"assist" : 13, "blocks" : 2, "submit\_date" : "2016-10-18"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "14" } }

{"team" : "Washington Wizards","name" : "John Wall","points" : 13,"rebound" : 2,"assist" : 12, "blocks" : 3, "submit\_date" : "2016-11-10"}

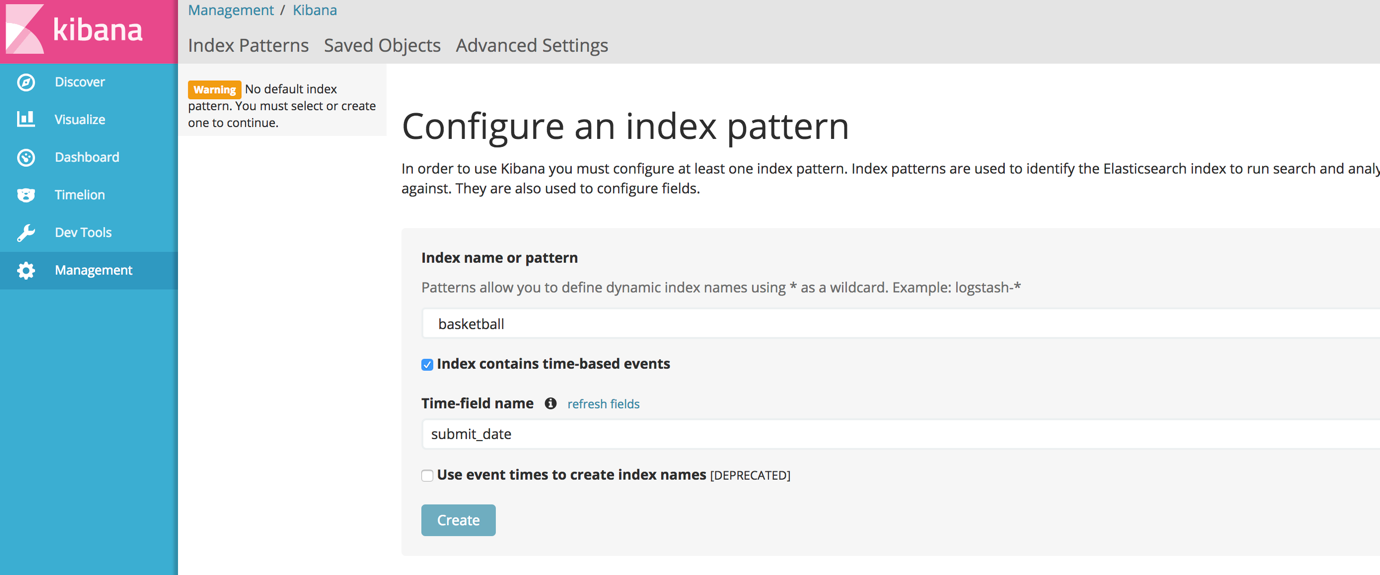
{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "15" } }

{"team" : "Washington Wizards","name" : "John Wall","points" : 15,"rebound" : 3,"assist" : 12, "blocks" : 3, "submit\_date" : "2016-11-22"}

{ "index" : { "\_index" : "basketball", "\_type" : "record", "\_id" : "16" } }

{"team" : "Washington Wizards","name" : "John Wall","points" : 22,"rebound" : 4,"assist" : 14, "blocks" : 3, "submit\_date" : "2016-11-27"}

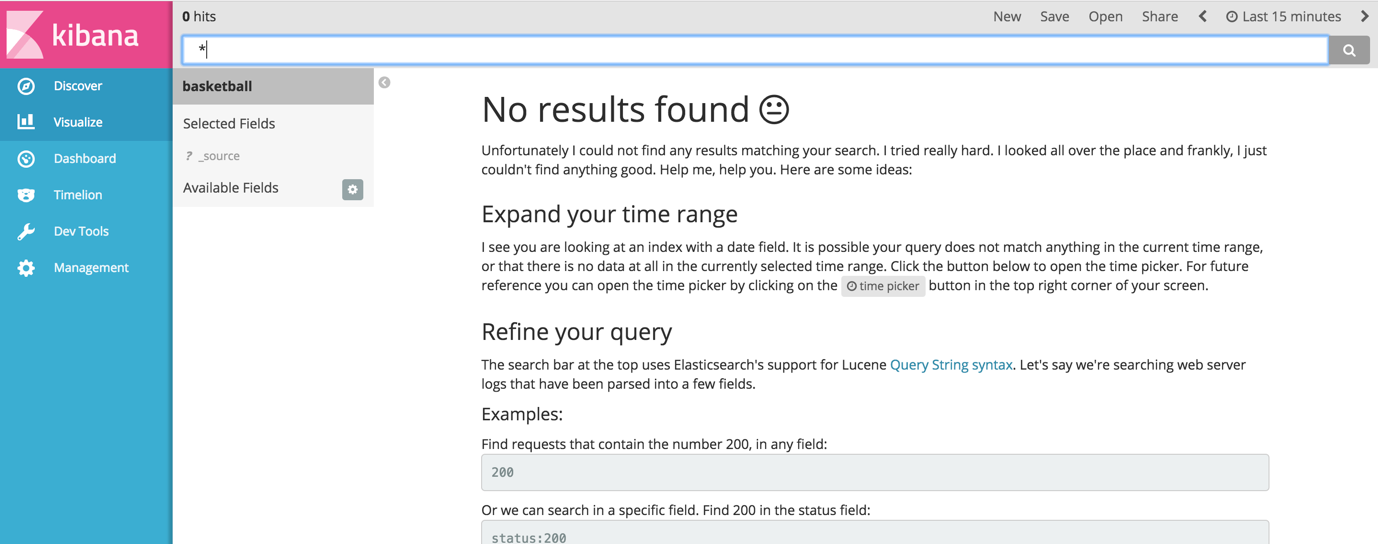
**Kibana의 Management – Index Patterns 접속**



* 인덱스 이름 입력하기
* 아까 넣어준 mapping에 의해 time-filed가 submit\_date라는 것을 kibana가 자동으로 알게 된다
* Create를 누르면 Elastic Search의 모든 값들을 kibana가 잘 이해하고 있는 것을 확인할 수 있다

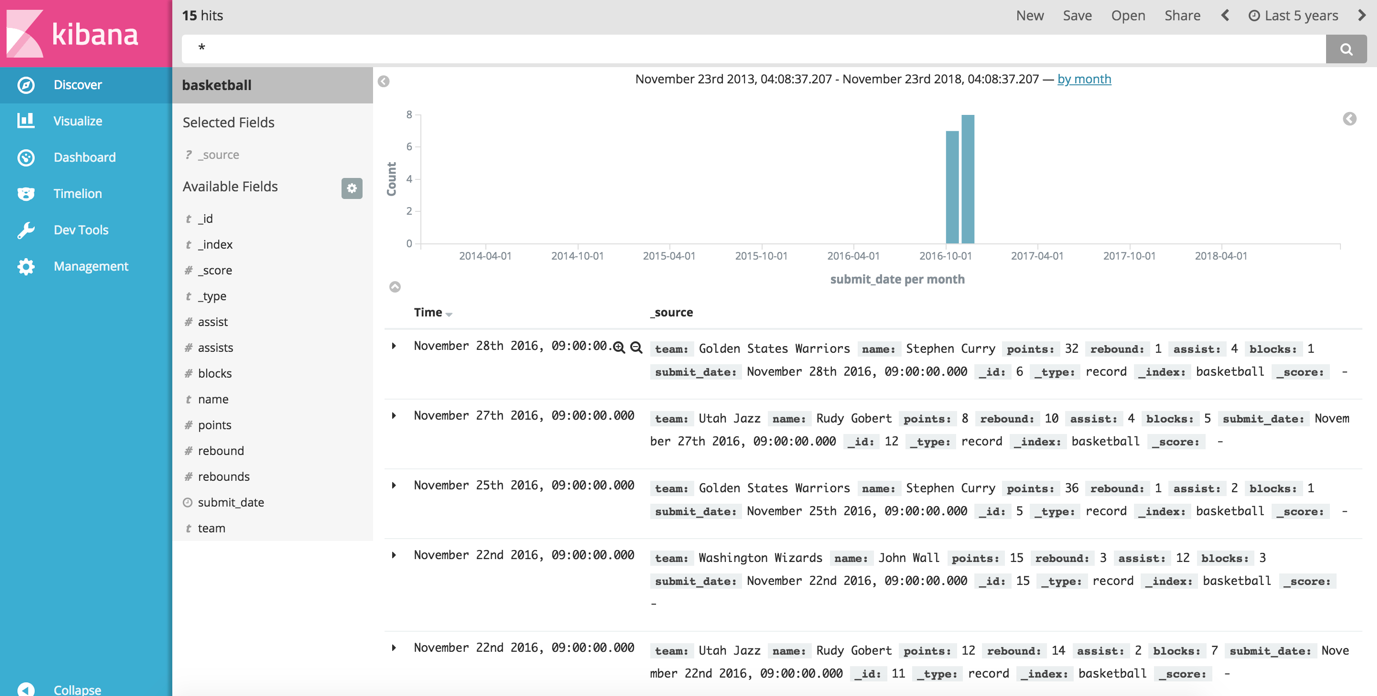
**Kibana Discover**

**Time Filter**



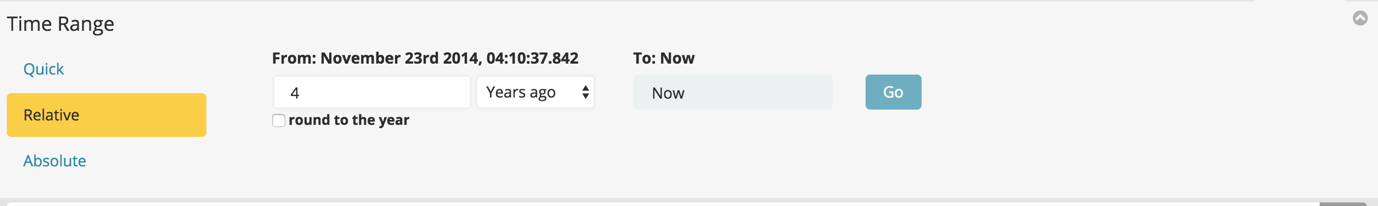
누르면 아무것도 안나온다. Time filter가 지난 15분이기 때문!!

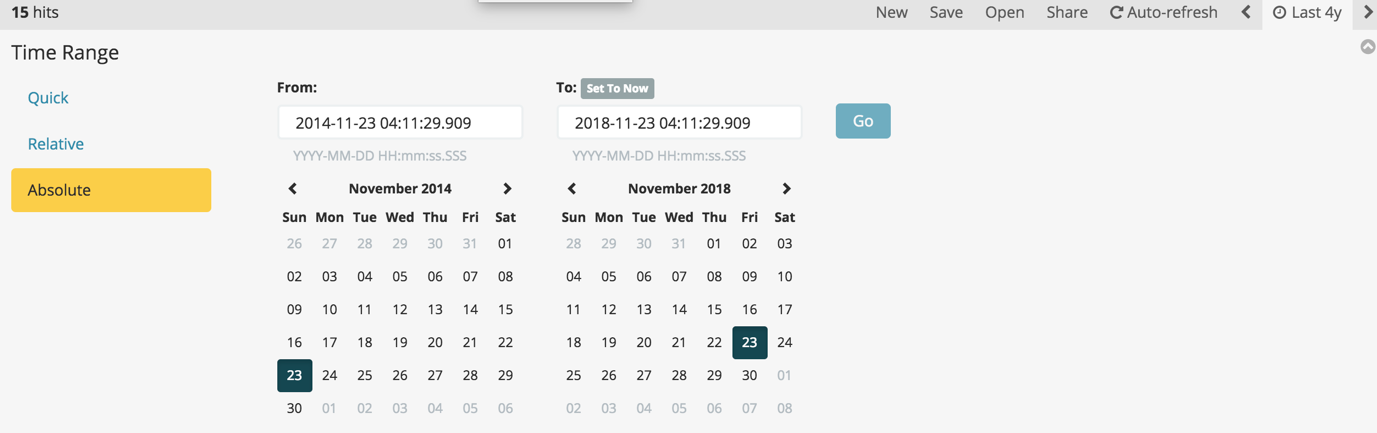
지난 5년으로 바꿔준다.



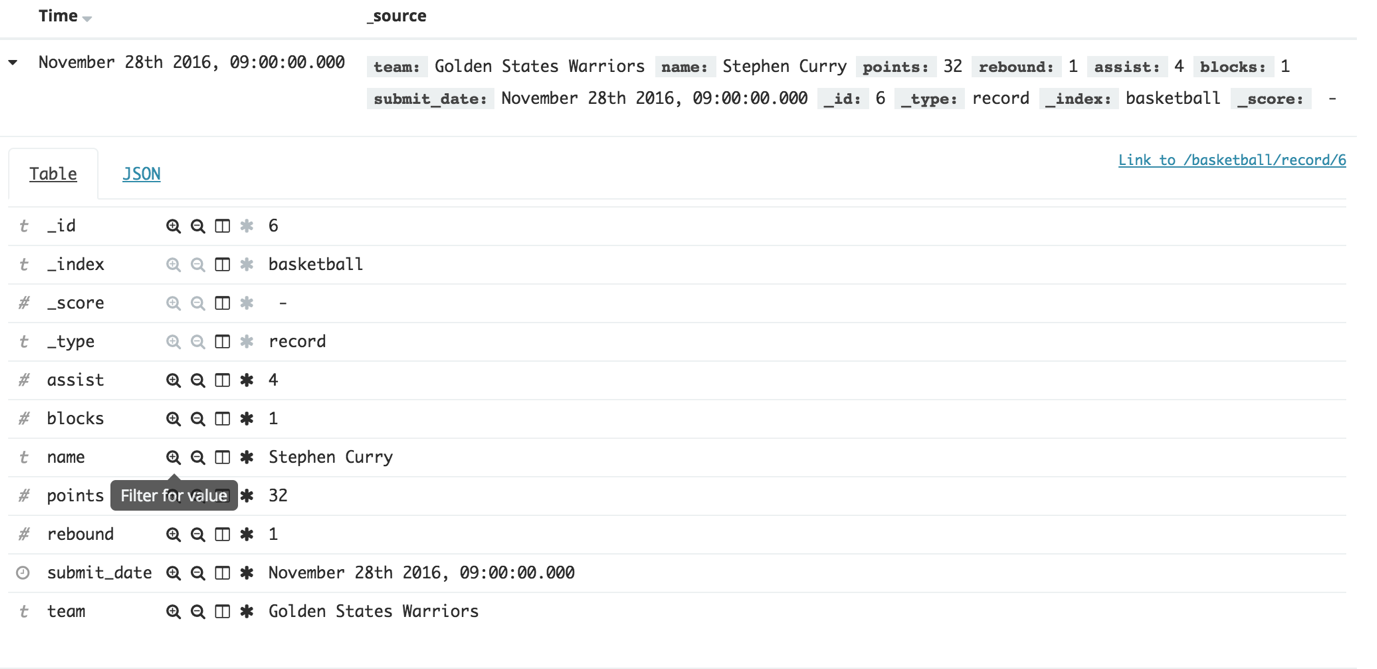
제대로 값이 나오는 모습을 확인할 수 있다! ☺

Time Range를 Relative, Absolute로도 확인할 수 있다



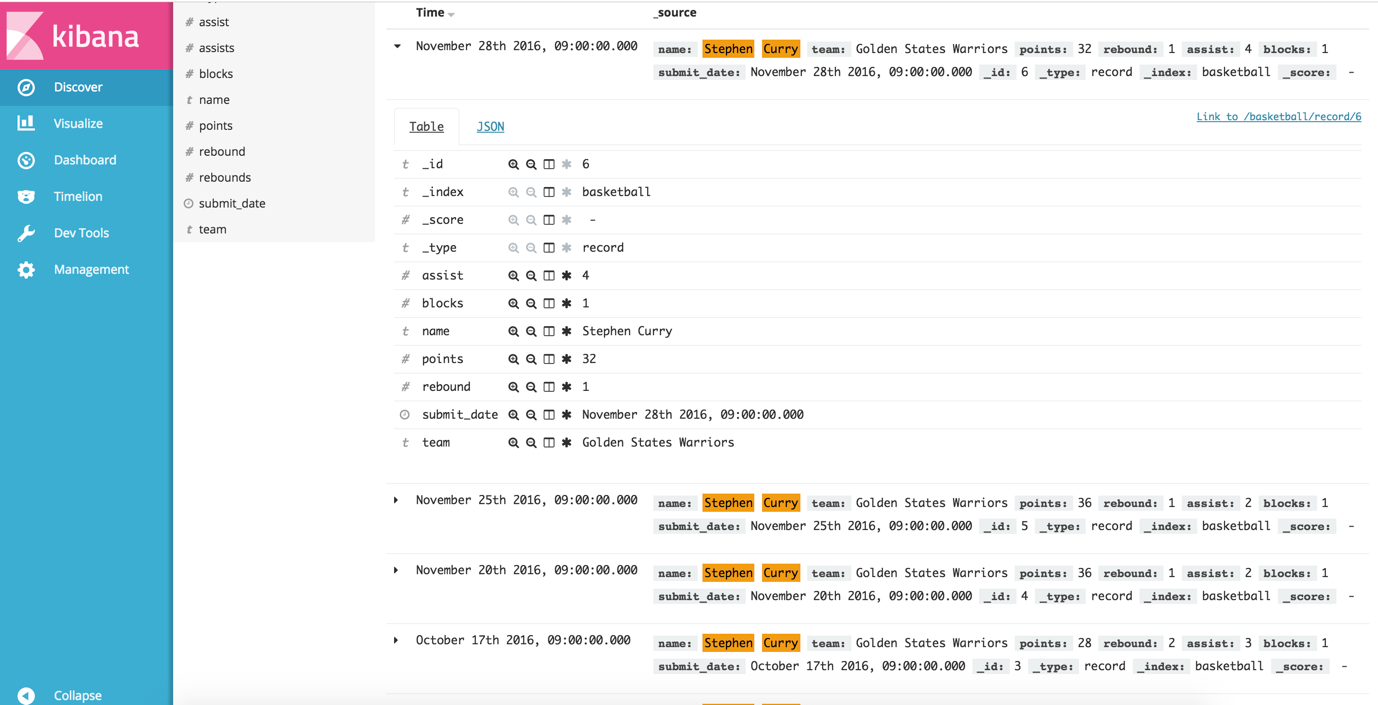


Document 하나에 대한 자세한 정보도 볼 수 있다.



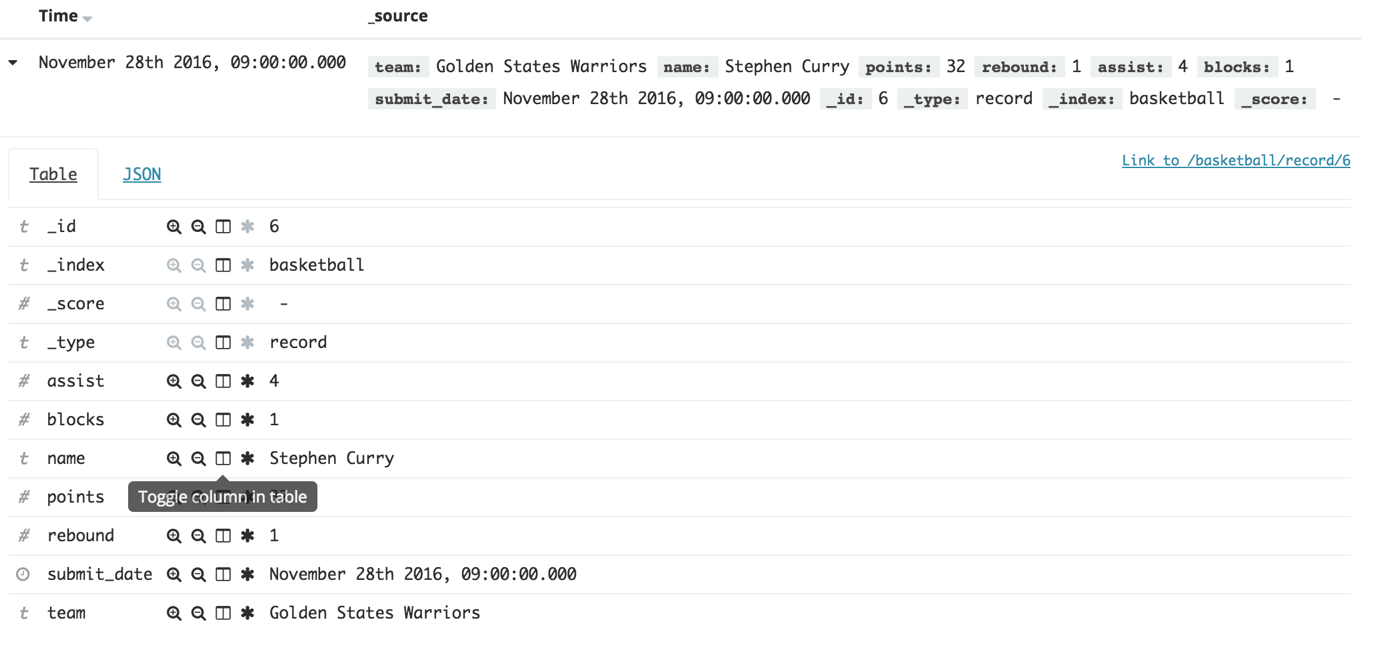
만약, name이 Stephen Curry인 Document만 보고싶다! 하면

저~기 위에처럼 클릭!하면 관련된 Document만 나온다.

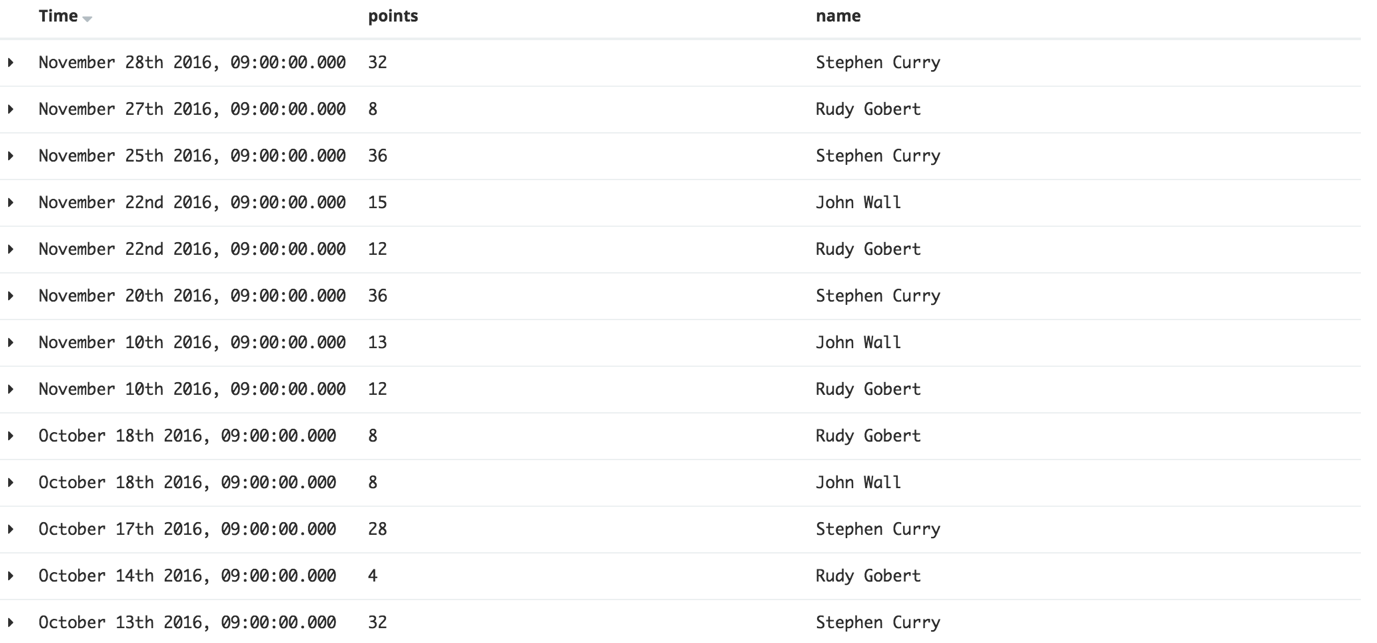


다시 모든 정보를 보고 싶으면 위에 휴지통 누른다

Toggle버튼으로 원하는 필드값만 볼 수 있다



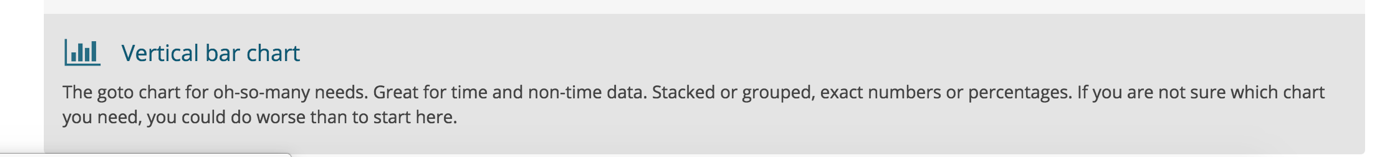
Name과 points 만 toggle버튼을 누르면, Table의 value에 2개만 나오는 모습!



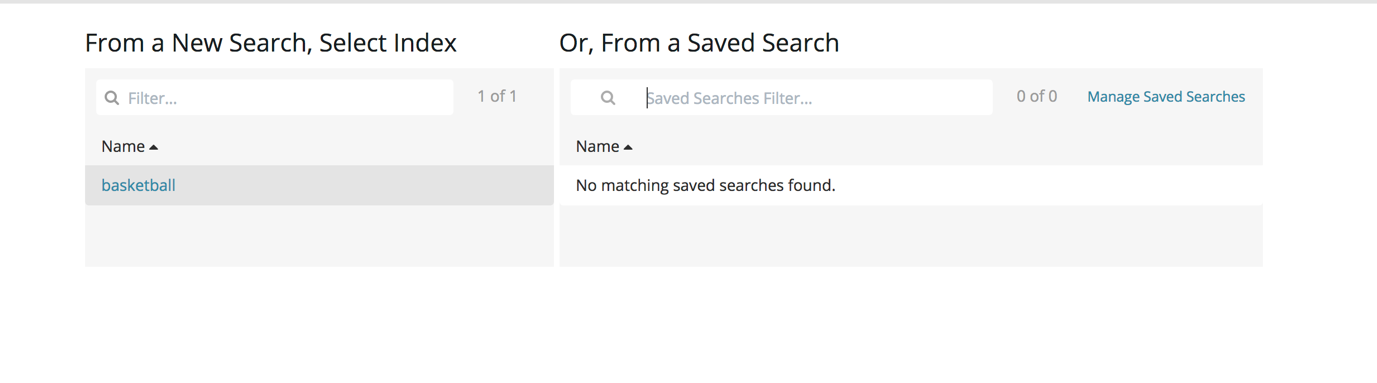
당연히 오름차순, 내림차순 정렬해서 값 확인도 가능!

**Kibana Visualize**

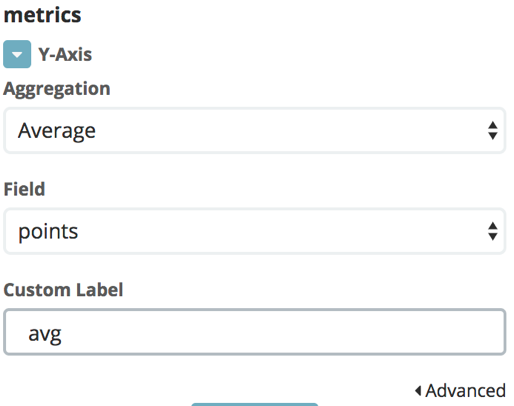
**막대 그래프**



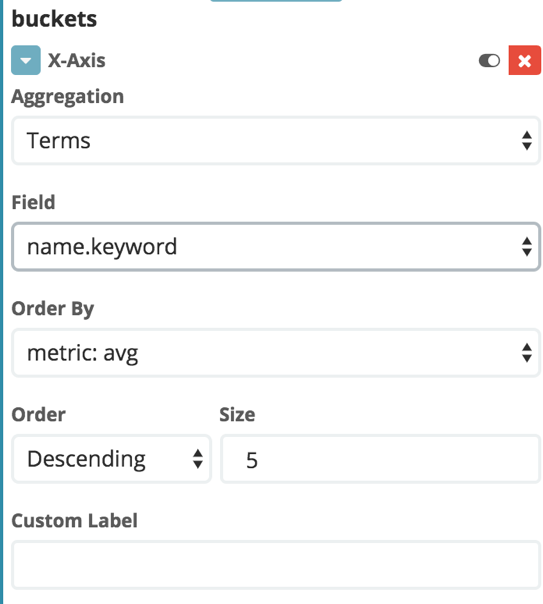
* 인덱스 선택



* 각 선수별 평균점수를 알아보자
* Y 축 지정 : points의 평균



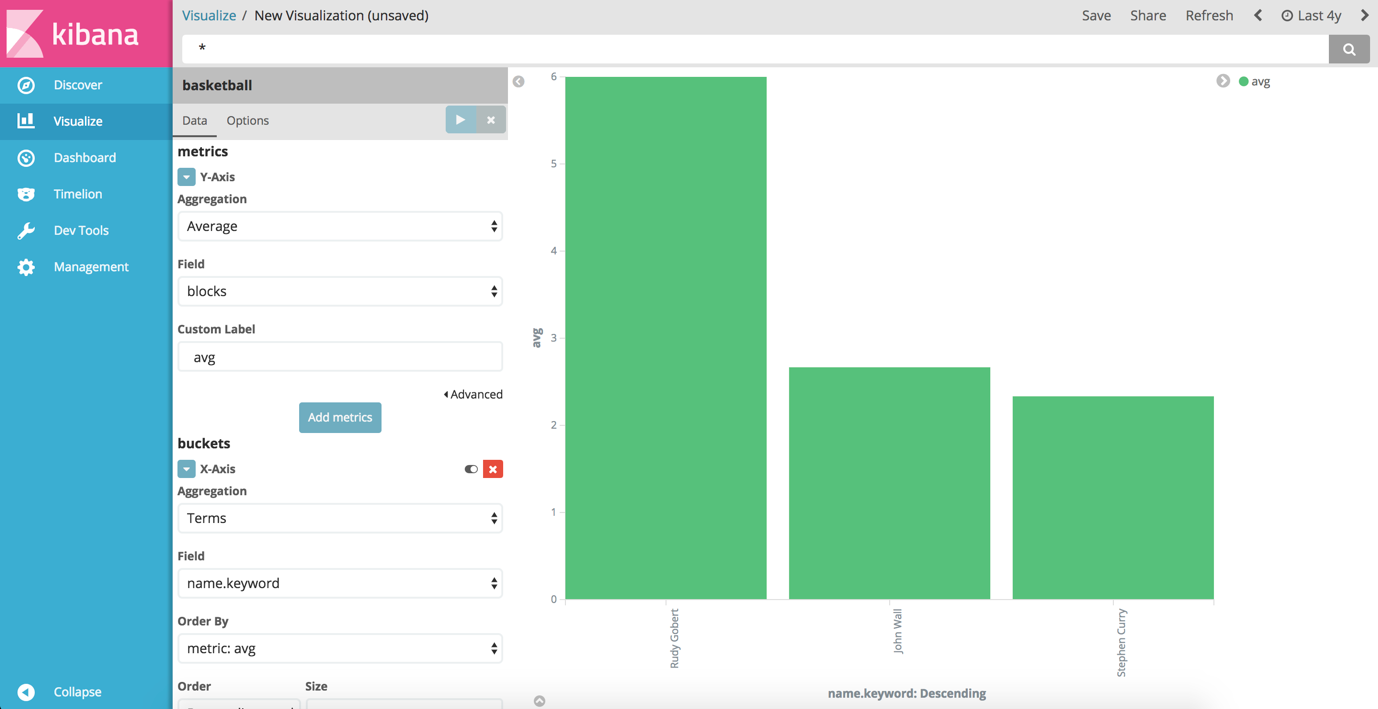
* X축 지정 : Term으로 묶기!!(선수 이름으로)



* 다음과 같은 결과를 볼 수 있다

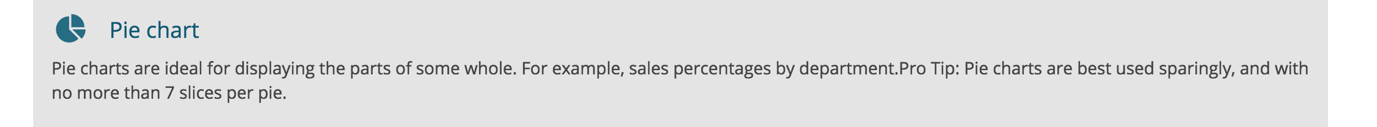


* 다른 예) 최고의 block 수를 갖는 선수는?

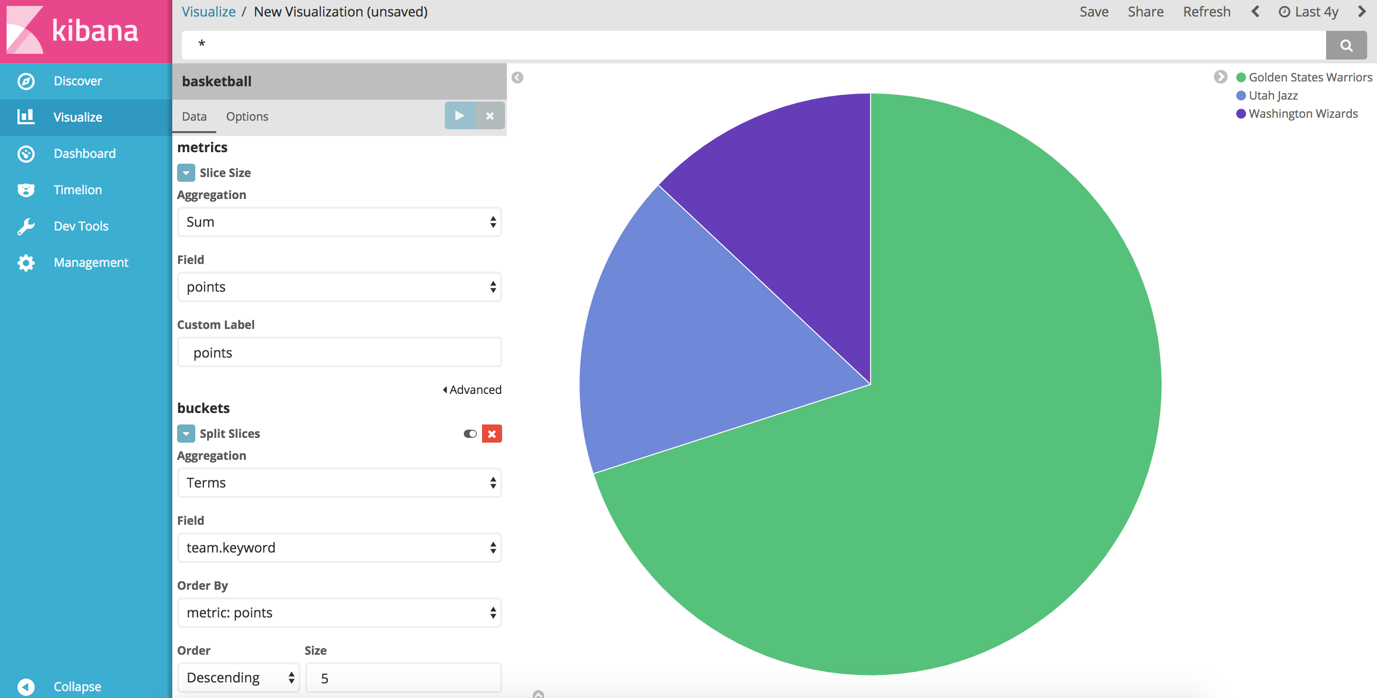


**Pie Chart**

* 선택



* 똑같이 인덱스 선택
* 각 팀의 점수 총합을 알아보자
* 파이차트는 1등, 2등, 3등을 알아낼 수 있다는 장점이 있다.



**타일 맵(지도에 표시) 예제**

마찬가지로, 인덱스 생성

**매핑 주기 – 매우 중요!! Geo-point**

curl -XPUT localhost:9200/classes/class/\_mapping -d @classesRating\_mapping.json

파일 내용)

{

"class" : {

"properties" : {

"title" : {

"type" : "string"

},

"professor" : {

"type" : "string"

},

"major" : {

"type" : "string"

},

"semester" : {

"type" : "string"

},

"student\_count" : {

"type" : "integer"

},

"unit" : {

"type" : "integer"

},

"rating" : {

"type" : "integer"

},

"submit\_date" : {

"type" : "date",

"format" : "yyyy-MM-dd"

},

"school\_location" : {

"type" : "geo\_point"

}

}

}

}

**GET으로 매핑 잘 들어갔는지 확인**

**Document들 추가 – bulk로**

curl -XPUT localhost:9200/\_bulk?pretty --data-binary @classes.json

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "1" } }

{"title" : "Machine Learning","Professor" : "Minsuk Heo","major" : "Computer Science","semester" : ["spring", "fall"],"student\_count" : 100,"unit" : 3,"rating" : 5, "submit\_date" : "2016-01-02", "school\_location" : {"lat" : 36.00, "lon" : -120.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "2" } }

{"title" : "Network","Professor" : "Minsuk Heo","major" : "Computer Science","semester" : ["fall"],"student\_count" : 50,"unit" : 3,"rating" : 4, "submit\_date" : "2016-02-02", "school\_location" : {"lat" : 36.00, "lon" : -120.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "3" } }

{"title" : "Operating System","Professor" : "Minsuk Heo","major" : "Computer Science","semester" : ["spring"],"student\_count" : 50,"unit" : 3,"rating" : 4, "submit\_date" : "2016-03-02", "school\_location" : {"lat" : 36.00, "lon" : -120.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "5" } }

{"title" : "Machine Learning","Professor" : "Tim Cook","major" : "Computer Science","semester" : ["spring"],"student\_count" : 40,"unit" : 3,"rating" : 2, "submit\_date" : "2016-04-02", "school\_location" : {"lat" : 39.00, "lon" : -112.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "6" } }

{"title" : "Network","Professor" : "Tim Cook","major" : "Computer Science","semester" : ["summer"],"student\_count" : 30,"unit" : 3,"rating" : 2, "submit\_date" : "2016-02-02", "school\_location" : {"lat" : 36.00, "lon" : -120.00}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "7" } }

{"title" : "Operating System","Professor" : "Jeniffer Anderson","major" : "Computer Science","semester" : ["winter"],"student\_count" : 30,"unit" : 3,"rating" : 1, "submit\_date" : "2016-11-02", "school\_location" : {"lat" : 39.97, "lon" : -89.78}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "8" } }

{"title" : "Algorithm","Professor" : "Tim Cook","major" : "Computer Science","semester" : ["fall"],"student\_count" : 80,"unit" : 3,"rating" : 2, "submit\_date" : "2016-10-22", "school\_location" : {"lat" : 39.97, "lon" : -89.78}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "9" } }

{"title" : "Data Structure","Professor" : "Tim Cook","major" : "Computer Science","semester" : ["winter"],"student\_count" : 50,"unit" : 3,"rating" : 2, "submit\_date" : "2016-07-22", "school\_location" : {"lat" : 39.97, "lon" : -89.78}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "10" } }

{"title" : "Computer Graphic","Professor" : "Jeniffer Anderson","major" : "Computer Science","semester" : ["spring"],"student\_count" : 60,"unit" : 2,"rating" : 3, "submit\_date" : "2016-11-12", "school\_location" : {"lat" : 39.97, "lon" : -89.78}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "11" } }

{"title" : "Music Fundamental","Professor" : "Jay Z","major" : "Music","semester" : ["fall"],"student\_count" : 100,"unit" : 3,"rating" : 5, "submit\_date" : "2016-05-22", "school\_location" : {"lat" : 42.51, "lon" : -74.83}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "12" } }

{"title" : "Vocal Techniques","Professor" : "Beyonce","major" : "Music","semester" : ["fall"],"student\_count" : 30,"unit" : 3,"rating" : 5, "submit\_date" : "2016-11-22", "school\_location" : {"lat" : 42.51, "lon" : -74.83}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "13" } }

{"title" : "Guitar Techiniques","Professor" : "Eric Clapton","major" : "Music","semester" : ["spring", "fall"],"student\_count" : 20,"unit" : 2,"rating" : 4, "submit\_date" : "2016-03-12", "school\_location" : {"lat" : 42.51, "lon" : -74.83}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "14" } }

{"title" : "Finance","Professor" : "Bill Gates","major" : "Accounting","semester" : ["winter"],"student\_count" : 50,"unit" : 3,"rating" : 2, "submit\_date" : "2016-01-12", "school\_location" : {"lat" : 42.51, "lon" : -74.83}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "15" } }

{"title" : "Marketing","Professor" : "Bill Gates","major" : "Accounting","semester" : ["spring"],"student\_count" : 60,"unit" : 2,"rating" : 3, "submit\_date" : "2016-01-22", "school\_location" : {"lat" : 42.51, "lon" : -74.83}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "16" } }

{"title" : "Accounting Information Systems","Professor" : "Tom Cruise","major" : "Accounting","semester" : ["fall"],"student\_count" : 100,"unit" : 2,"rating" : 1, "submit\_date" : "2016-11-12", "school\_location" : {"lat" : 42.51, "lon" : -74.83}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "17" } }

{"title" : "Individual Taxation","Professor" : "Tom Cruise","major" : "Accounting","semester" : ["fall"],"student\_count" : 30,"unit" : 1,"rating" : 2, "submit\_date" : "2016-08-02", "school\_location" : {"lat" : 42.32, "lon" : -94.74}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "18" } }

{"title" : "Auditing","Professor" : "Victoria Park","major" : "Accounting","semester" : ["spring", "fall"],"student\_count" : 20,"unit" : 2,"rating" : 3, "submit\_date" : "2016-09-13", "school\_location" : {"lat" : 42.32, "lon" : -94.74}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "19" } }

{"title" : "Cell Biology","Professor" : "Anjella Kim","major" : "Medical","semester" : ["fall"],"student\_count" : 40,"unit" : 5,"rating" : 5, "submit\_date" : "2016-02-22", "school\_location" : {"lat" : 42.32, "lon" : -94.74}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "20" } }

{"title" : "Physiology","Professor" : "Jack Berk","major" : "Medical","semester" : ["summer"],"student\_count" : 30,"unit" : 5,"rating" : 4, "submit\_date" : "2016-11-12", "school\_location" : {"lat" : 32.69, "lon" : -99.44}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "21" } }

{"title" : "Neuroscience","Professor" : "Jihee Yang","major" : "Medical","semester" : ["spring", "fall"],"student\_count" : 20,"unit" : 5,"rating" : 4, "submit\_date" : "2016-06-03", "school\_location" : {"lat" : 32.69, "lon" : -99.44}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "22" } }

{"title" : "Immunology","Professor" : "Meredith Lee","major" : "Medical","semester" : ["winter"],"student\_count" : 30,"unit" : 3,"rating" : 2, "submit\_date" : "2016-06-21", "school\_location" : {"lat" : 32.69, "lon" : -99.44}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "23" } }

{"title" : "Genetics","Professor" : "David Pollack","major" : "Medical","semester" : ["spring"],"student\_count" : 20,"unit" : 3,"rating" : 3, "submit\_date" : "2016-06-30", "school\_location" : {"lat" : 28.22, "lon" : -81.87}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "24" } }

{"title" : "Biochemistry","Professor" : "John Miller","major" : "Medical","semester" : ["fall"],"student\_count" : 30,"unit" : 3,"rating" : 4, "submit\_date" : "2016-01-11", "school\_location" : {"lat" : 28.22, "lon" : -81.87}}

{ "index" : { "\_index" : "classes", "\_type" : "class", "\_id" : "25" } }

{"title" : "Anatomy","Professor" : "Tom Johnson","major" : "Medical","semester" : ["fall"],"student\_count" : 30,"unit" : 5,"rating" : 3, "submit\_date" : "2016-11-12", "school\_location" : {"lat" : 28.22, "lon" : -81.87}}

**GET으로 id별로 Document 잘 들어갔는지 확인**

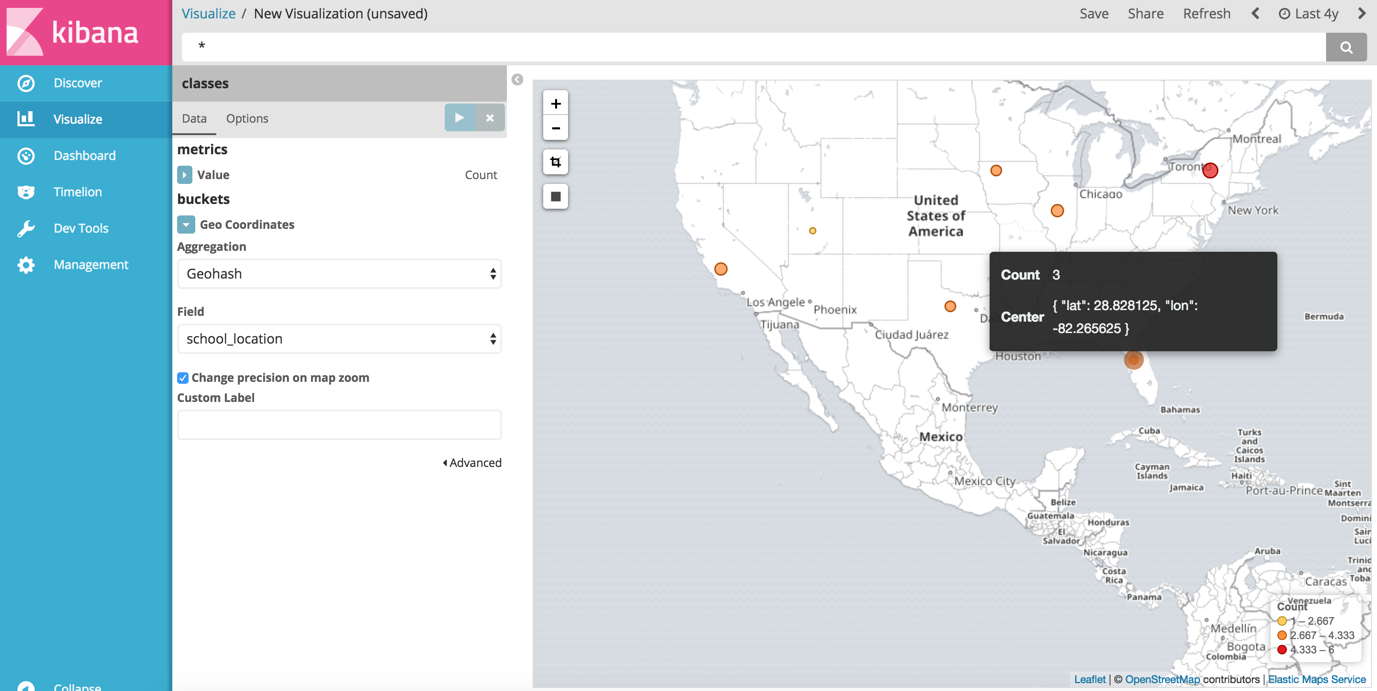
curl -XGET localhost:9200/classes/class/1?pretty

**Kibana에서 확인**

1) Management에서 Classes 인덱스 선택

2)Visualize에서 Tile map 선택

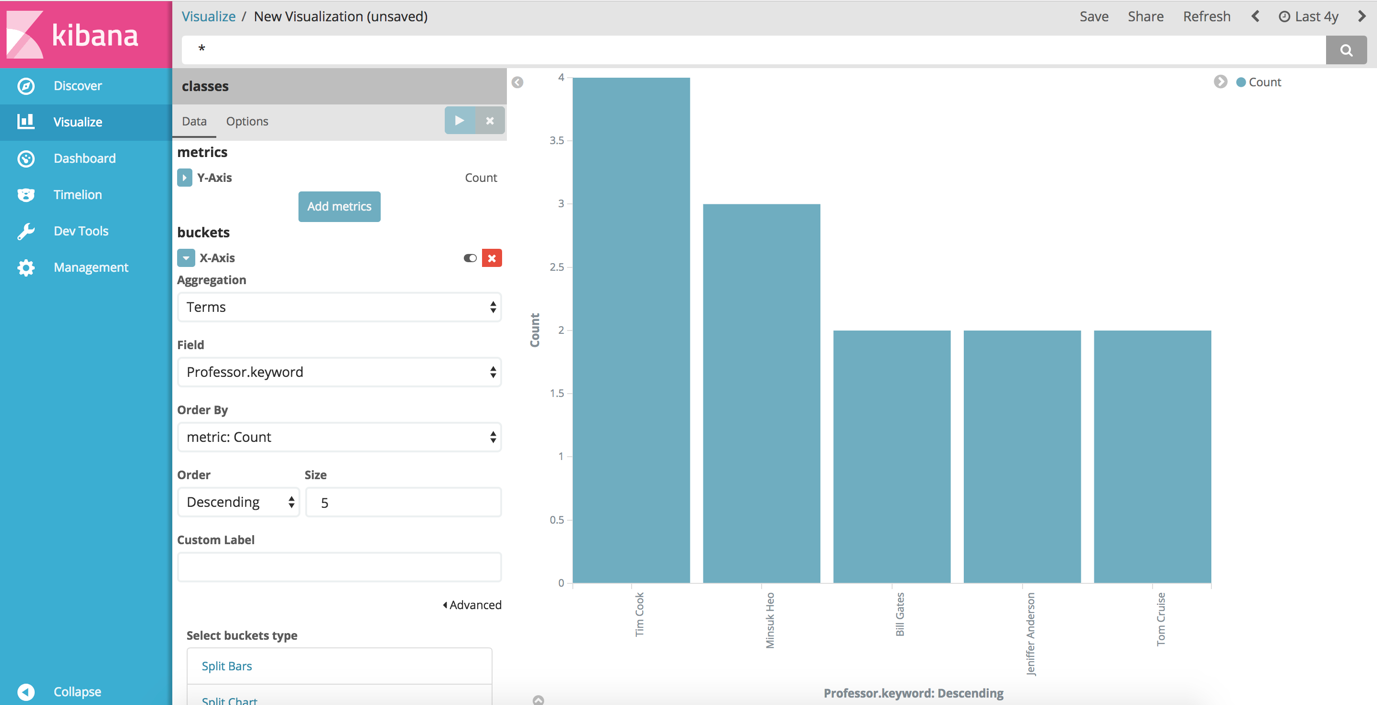
3)buckets를 Geo Coordinates로 선택하고 맞는 Field 선택



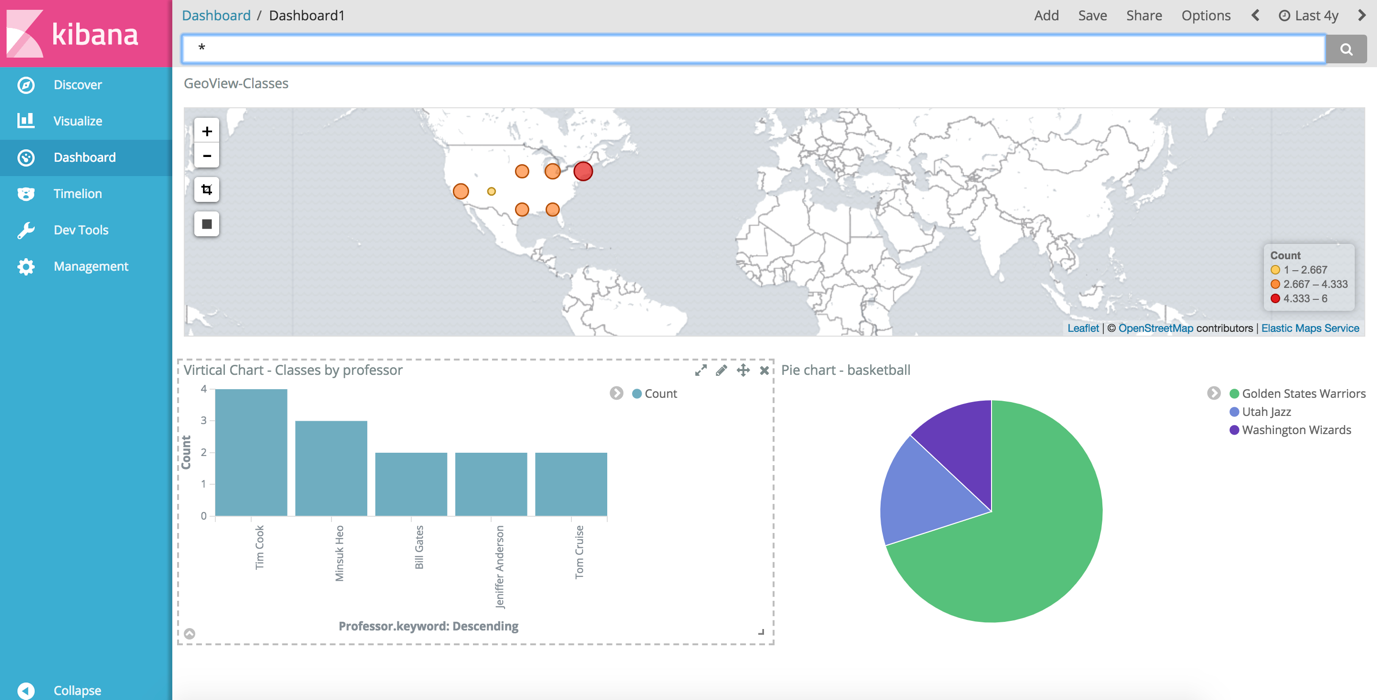
**Kibana Dashboard**

* Visualize에서 Chart만들어서 Save한다

예제에서는 교수님들의 강의 평가의 Virtical chart를 Save한다

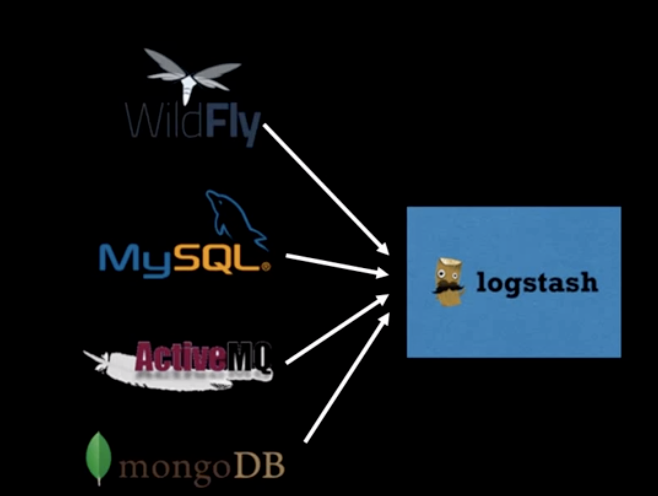


* Visualize에서 저장한 Chart들은 Dashboard에서 추가해서 볼 수 있다.



**Logstash**

* ELK 스택에서 logstash는 input을 담당한다
* Logstash -> Elastic Search -> Kibana
* Logstash는 세상의 많은 종류의 데이터를 자기껄로 흡수해서, 변환시켜서 Elastic Search에 보낼 수 있다



**설치하기**

wget <https://artifacts.elastic.co/downloads/logstash/logstash-5.3.1.deb>

sudo dpkg -i logstash-5.3.1.deb

* Install path: /usr/share/logstash

**Config 수정**

경로 이동

cd /usr/share/logstash

아래와 같은 Config 파일을 생성한다.

sudo vi logstash-simple.conf

내용은 이렇게 입력

input {

stdin { }

}

output {

stdout { }

}

**실행시키기**

sudo /usr/share/logstash/bin/logstash -f ./logstash-simple.conf

내가 만든 conf 파일대로 Logstash가 실행되는 모습을 확인할 수 있다

**실전 ELK 활용 실데이터 분석 1**

**Step1. 실 데이터 가져오기**

#### Datas site

<https://catalog.data.gov/dataset>

#### Population analysis Datas

#### <https://catalog.data.gov/dataset/population-by-country-1980-2010>

#### csv파일

#### 데이터 다운 받기

Site에서 받은 데이터는 약간의 수정이 필요하다. 아래의 데이터는 바로 사용할 수 있는 데이터.

wget <https://raw.githubusercontent.com/minsuk-heo/BigData/master/ch06/populationbycountry19802010millions.csv>

**Step2. Elasticsearch와 Kibana가 잘 켜져있나 확인**

ps -ef | grep kibana

Kibana 확인)

##### **Running인 경우**

root 29968 29933 9 16:58 pts/0 00:00:06 /usr/share/kibana/bin/../node/bin/node --no-warnings /usr/share/kibana/bin/../src/cli

root 30036 30018 0 16:59 pts/1 00:00:00 grep --color=auto kibana

##### **Stopped인 경우**

root 29957 29933 0 16:57 pts/0 00:00:00 grep --color=auto kibana

**Kibana 켜기**

sudo /usr/share/kibana/bin/kibana

ElasticSearch 확인)

service elasticsearch status

또는 GET 날려서 확인

켜기)

sudo service elasticsearch start

**Step3. Logstash config하기**

**Config 수정**

경로 이동

cd /usr/share/logstash

아래와 같은 Config 파일을 생성한다.

sudo vi logstash.conf

내용은 이렇게 입력

input {

file {

path => "/home/minsuk/Documents/git-repo/BigData/ch06/populationbycountry19802010millions.csv"

start\_position => "beginning"

sincedb\_path => "/dev/null"

}

}

filter {

csv {

separator => ","

columns => ["Country","1980","1981","1982","1983","1984","1985","1986","1987","1988","1989","1990","1991","1992","1993","1994","1995","1996","1997","1998","1999","2000","2001","2002","2003","2004","2005","2006","2007","2008","2009","2010"]

}

mutate {convert => ["1980", "float"]}

mutate {convert => ["1981", "float"]}

mutate {convert => ["1982", "float"]}

mutate {convert => ["1983", "float"]}

mutate {convert => ["1984", "float"]}

mutate {convert => ["1985", "float"]}

mutate {convert => ["1986", "float"]}

mutate {convert => ["1987", "float"]}

mutate {convert => ["1988", "float"]}

mutate {convert => ["1989", "float"]}

mutate {convert => ["1990", "float"]}

mutate {convert => ["1991", "float"]}

mutate {convert => ["1992", "float"]}

mutate {convert => ["1993", "float"]}

mutate {convert => ["1994", "float"]}

mutate {convert => ["1995", "float"]}

mutate {convert => ["1996", "float"]}

mutate {convert => ["1997", "float"]}

mutate {convert => ["1998", "float"]}

mutate {convert => ["1999", "float"]}

mutate {convert => ["2000", "float"]}

mutate {convert => ["2001", "float"]}

mutate {convert => ["2002", "float"]}

mutate {convert => ["2003", "float"]}

mutate {convert => ["2004", "float"]}

mutate {convert => ["2005", "float"]}

mutate {convert => ["2006", "float"]}

mutate {convert => ["2007", "float"]}

mutate {convert => ["2008", "float"]}

mutate {convert => ["2009", "float"]}

mutate {convert => ["2010", "float"]}

}

output {

elasticsearch {

hosts => "localhost"

index => "population"

}

stdout {}

}

conf파일 내용 설명)

**input**

Start\_position 의 default값은 end(보통 streaming으로 데이터가 들어오니까 end이다. 그런데 지금은 tutorial 이니까 파일의 처음부터 데이터를 읽자.)

Sincedb\_path : 한 번 들어간 데이터를 logstash 가 넣지 않기 때문에, /dev/null로 설정 해줘야 계속 logstash에 데이터가 잘 들어간다.

**Filter**

Mutate : 데이터를 숫자로 받아들이기 위해 float로 표시함

**Output**

Elasticsearch{

Host => , Index=> 지정

}

**Step4. Logstash를 conf파일과 함께 돌린다 => ElsasticSearch에 잘 들어간다**

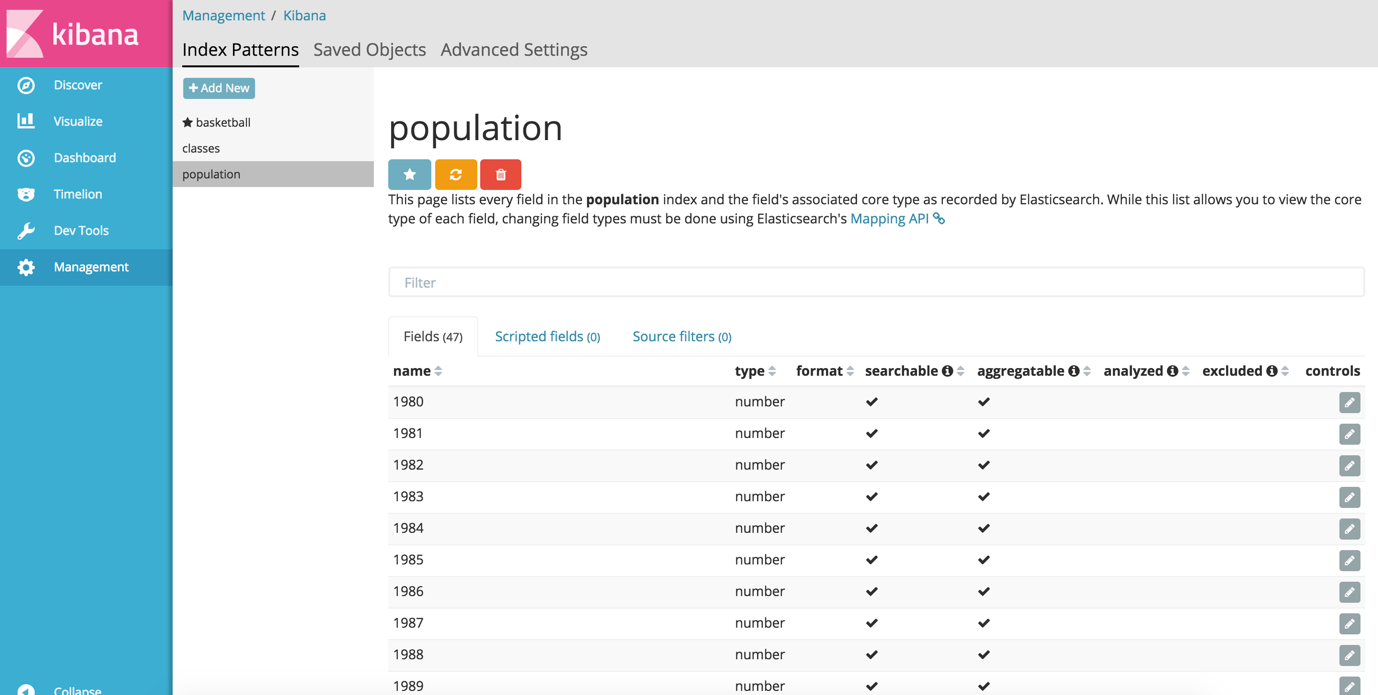
sudo /usr/share/logstash/bin/logstash -f ./logstash.conf

**Step5. Kibana 접속**

Ip주소:5601로 브라우저에서 접속

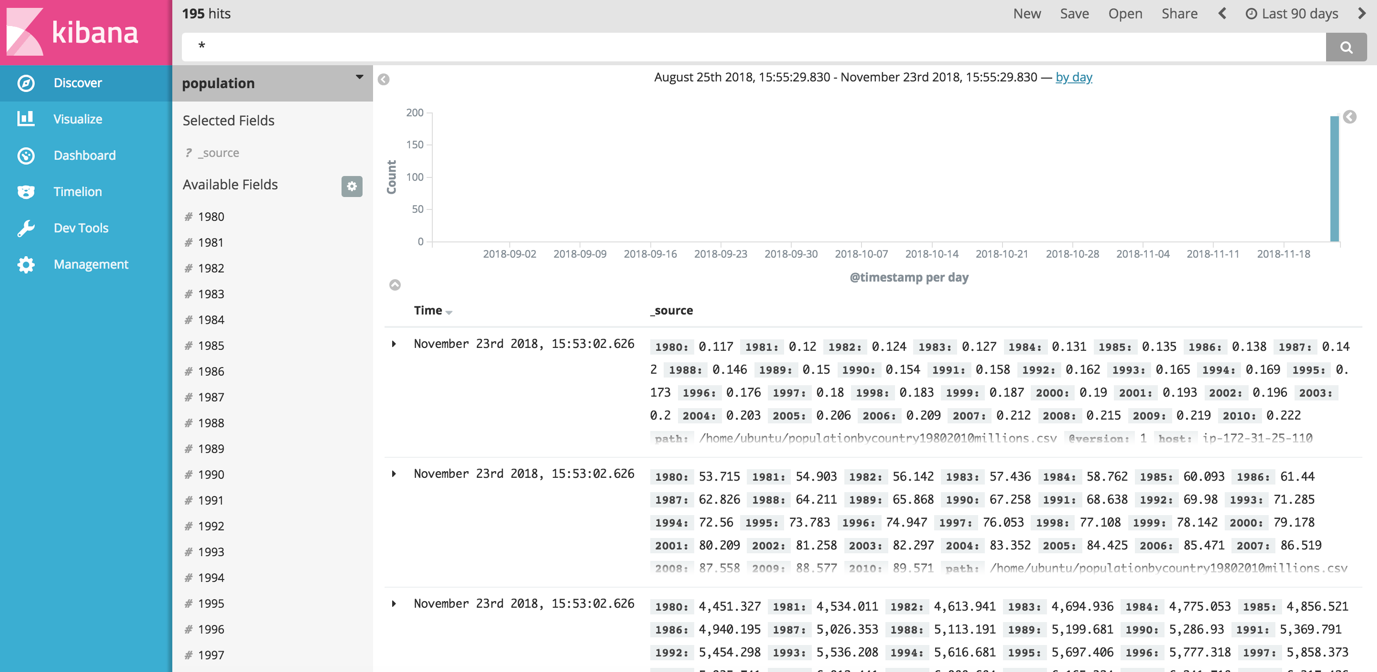
* Management 확인

Add Index – population 입력 – Create



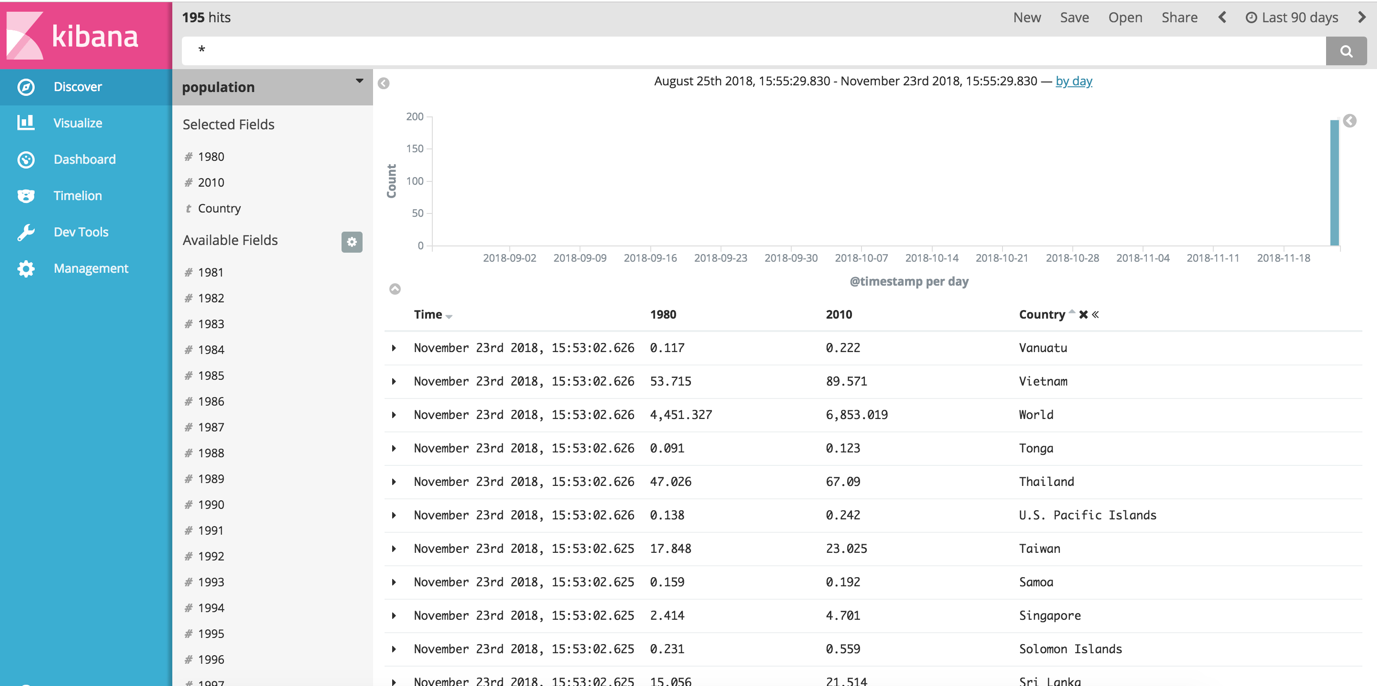
* Discover에서 데이터 확인

Population선택

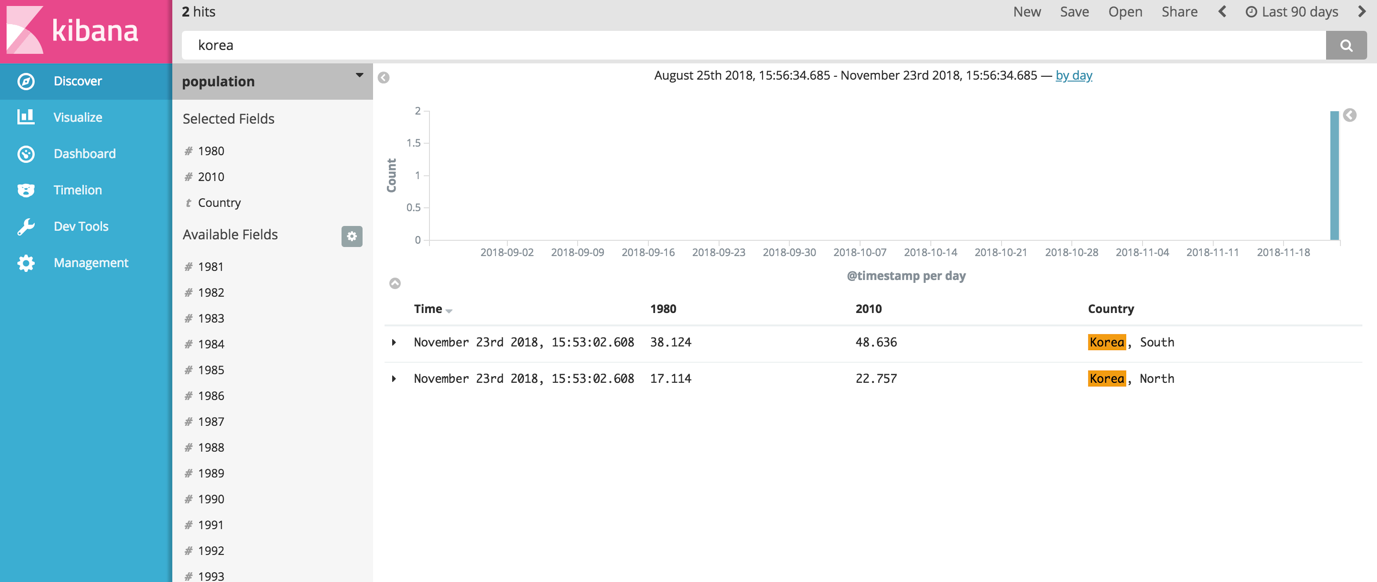


Document 하나하나 클릭해서 확인

원하는 정보만 Toggle – County, 1980, 2010 => Table View에서 깔끔하게 확인



Search해보기 – korea, United States 등 검색



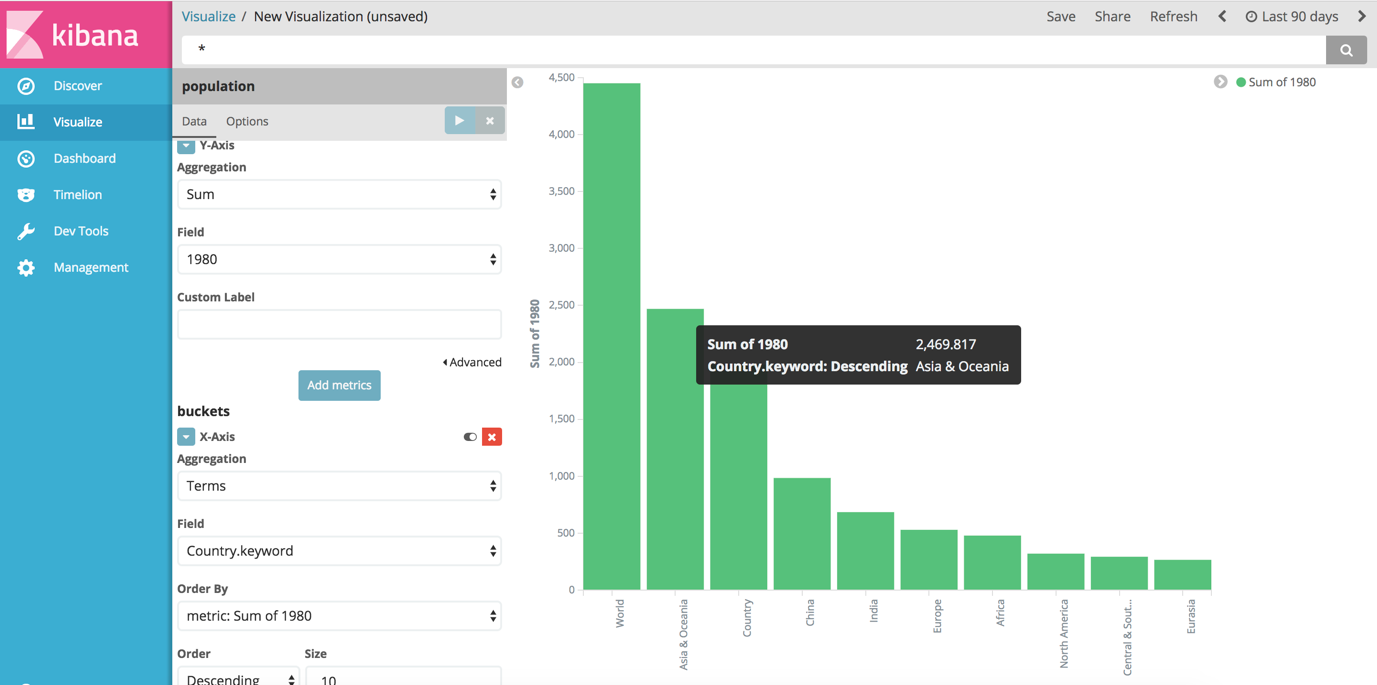
* Visualize

차트 만들기 – Vertical var chart

Sum, 1980 인구수 확인

Term은 나라별로(county), size=10으로 해서 10개의 나라만 보자

2010년 인구도 동일하게 확인



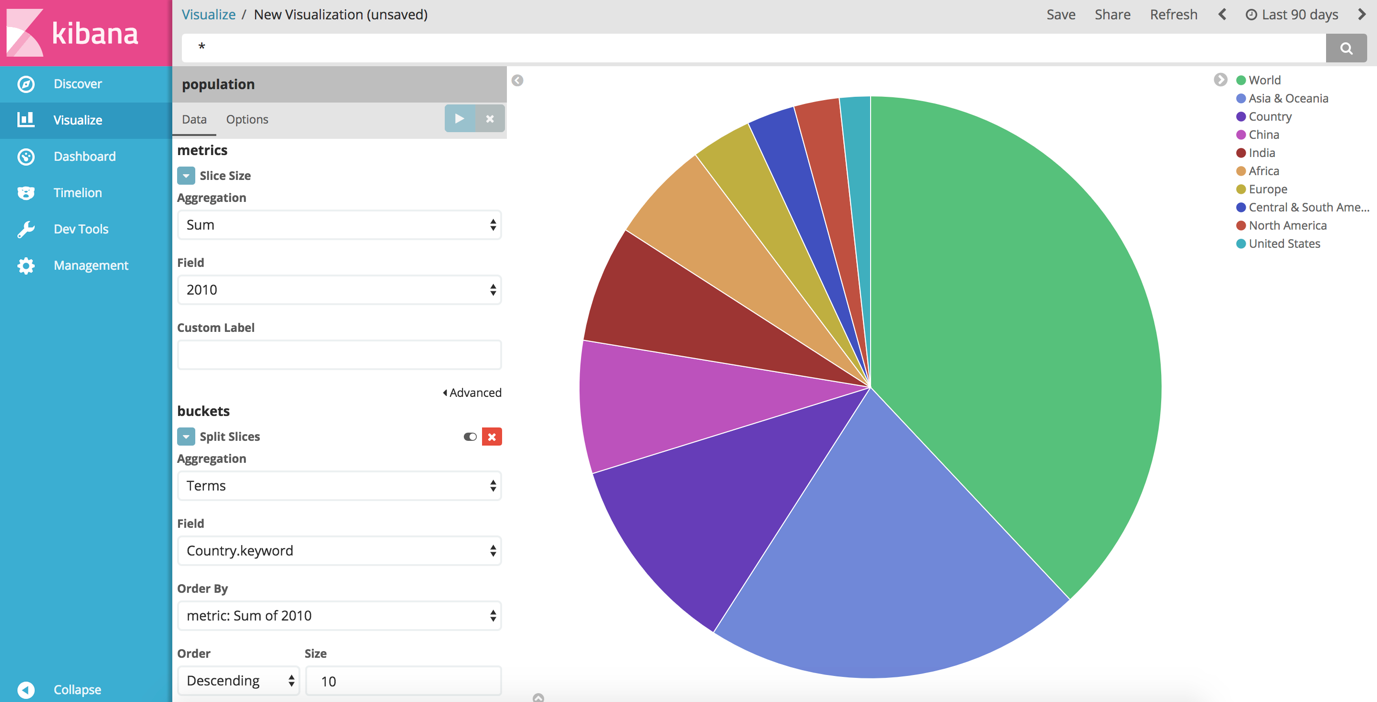
Save 한다

차트 만들기 – Pie chart

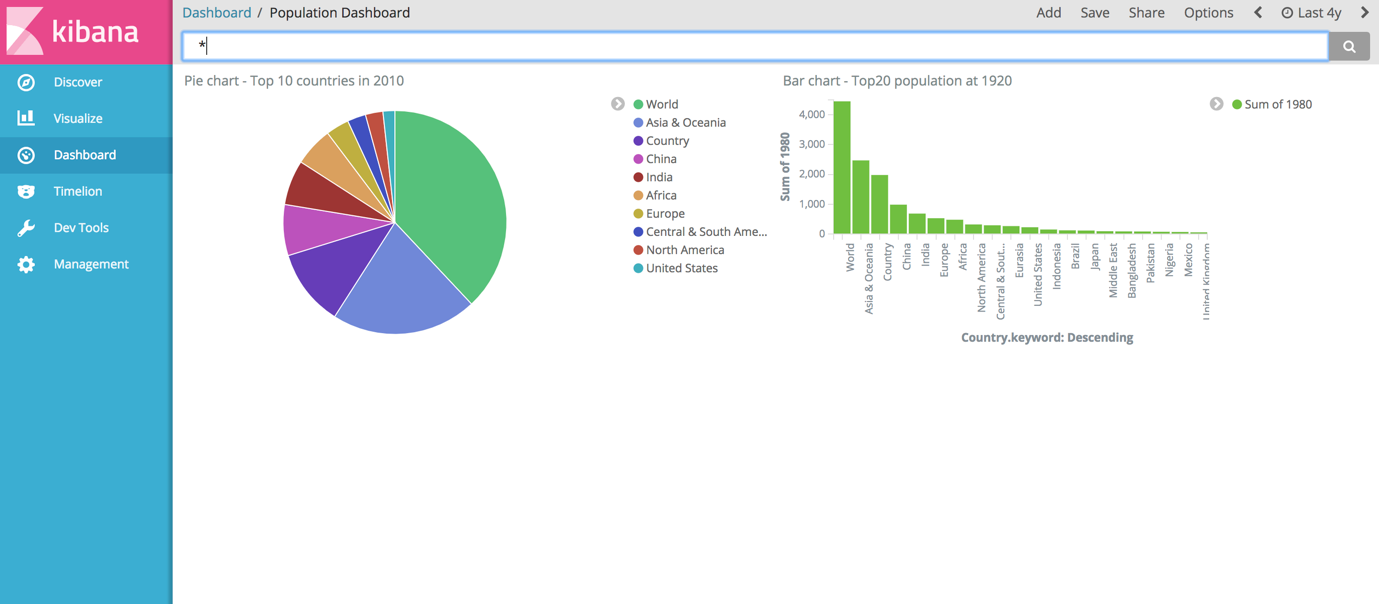
Sum, 1980

Bucket은 term으로 county

Size를 10으로 바꾸면 10개의 나라



* Dashboard에 저장하기



**실전 ELK 활용 실데이터 분석 2**

Step0. Data 가져오기

주식 데이터

Yahoo finance

FB검색

Historical data눌러서, 5년동안 주식 정보 Apply 누르고, Download Data(csv파일)

Step1. 엘라스틱 서치, 키바나 잘 돌아가나 확인

Step2. Logstash cofig하기

설명 )

위와 거의 동일

Step3. Conf 파일과 함께 logstash 돌리기 => Elastic Search에 데이터 삽입

Step4. Kibana에서 확인

주소:5601

* Management에 Index 추가

Stock, time에는 Date

잘 들어간 것 확인

* Discover에서 확인

인덱스 선택

Time 지난 5년으로 선택

Document 값 Toggle – 시간, open, close

Discover를 Save하기(나중에 Dashboard에서 이용)

* Visualize

차트 그리기 – Line Graph

Y축은 평균, close 값

X축은 날짜

저장하기

차트 그리기 – Metric

몇개의 데이터가 있나 확인(document수)

* Dashboard

차트 추가하고,

Discover도 추가!!

**클라우드 로그 분석 시스템 아키텍처**

1) Elasticsearch : 로그를 저장하는 데이터 베이스

2) Logstash : 클라우드 내에서 발생된 로그를 전달받아서 엘라스틱 서치에 저장

3) Kibana : ElasticSearch에 저장된 로그를 브라우저에 출력

로그스태시는 어떻게 클라우드 상의 로그 정보를 입력받을 수 있을까?

정답은 FileBeat!

FileBeat는 각각의 서버에 설치되어서, 로그파일에 변화가 있을 떄, 변화된 로그파일을 로그스태시로 전송해준다.

오래된 데이터 삭제 문제 – curator 로 해결

한달 이상 된 데이터를 삭제할건데, 나중에 필요할 수 있으니까 => S3에 저장(매일 발생한 로그)

**FileBeat**

**설치하기**

**FileBeat설정하기 – filebeat.yml**

파일 내용) 아래 2군데 변경

Input type log : 로그 파일 경로

Output – logstash : logstash IP 넣어주기

**Logstash 설정하기**

파일 내용)

Input부분 : beats에서 받겠다

Output 부분 : hostname별로 (웹서버, 디비 등등 서버 주소별로) elasticsearch 설정 가능. 각각 인덱스 다르게

**변경된 설정 적용 – 서비스 재시작**

Logstash 재시작

서버로 가서 FileBeat 재시작

Elastic Search 재시작

Kibana 재시작

**Kibana에서 확인**

**Curator**

* 서버 디스크 공간의 한계가 있기 때문에, 한달 이상된 인덱스는 지워라, 디스크 사용랑 75퍼면

**설치하기**

Pip install elasticsearch-curator

**설치 잘 되었나 확인**

Curator명령어

**Curoator config하기**

Curator.yml 파일

**Time base(기준) Delete**

Delete\_indices\_time\_base.yml

Delete\_indices\_time\_base.sh

파일 내용)

30일 기준으로 설정, “tomcat”으로 시작하는 인덱스를, 30일이 지나면 삭제해라

쉘 스크립트 파일 내용)

**Data base(기준) Delete**

Delete\_indeices\_size\_base.yml

Delete\_indeices\_size\_base.sh

어느정도 디스크 사용량이 되면 오래된 인덱스부터 지워라

파일 내용)

300GB로 설정 (disk\_space)

1과 2, 2개(Index 갯수만큼 추가!!)

**손으로 shell script 돌리지말고, Curator를 시스템 자체적으로 돌린다. => 크론탭!!!사용!!**

cd /etc/cron.d

curator\_cron

daily로 Job 지정해서 Shell 스크립트 넣어 줌

**S3에 백업하기**

**Step0. S3 Repository plugin 설치하기**

* 설치
* 하고 나서 엘라스틱서치 재시작

**Step1. S3 백업을 curl 커맨드로 등록해준다**

* Access\_key, scret\_key, bucket, region 정보 알고 있어야 함

Step2. 백업 매일 하기 – Crontab

Shell 스크립트 추가(Backup script)

내용 설명) 어제 날짜를 입력 받고, 모든 로그 선언

* 백업 잘 저장 되었나 확인

Curl -XGET \_snapshot 확인 !!!!

Step3. Kibana에서 확인

데이터 삭제해보고, S3에 있는 인덱스로 복원해보자

삭제 커맨드 날린다. 2017-08-06인덱스를 삭제한다

다시 키바나에 접근 해보면, 당연히 데이터가 보이지 않는다. 인덱스가 사라졌으니까

다시 복원하기!! -> 스크립트 이용

Elk\_restore.sh 복원하고싶은 인덱스 이름

다시 Kibana에서 확인