Elastic Stack을 활용한 Data Dashboard 만들기

Week 6 - Dashboard 만들기 최종실습

Fast Campus

1. 전체 Review

지금까지 Elastic Stack에 대해 배우긴 했는데

실제로 처음부터 끝까지 어떻게 어떤 순서로 작업해야 되는지 모르겠다.

지금까지 배운 걸 어떻게 이용하면 되는지 순서대로 살펴보자.

Elastic Stack을 실행할 Server가 필요하다 👑

(Elasticsearch node 1대 기준)

	권장 👑	수업 실습 서버
Memory	16GB - 64GB	8GB
CPUs	2 - 8 core	2 core
Disk	SSD	SSD

위를 참고해서 적절한 hardware를 선택하자

Elastic Stack을 설치/실행하자 👑

1. zip/tar.gz 👑
2. debian package 👑
3. RPM repository 👑
4. Windows msi (beta) 👑
5. Docker 👑

각자의 개발환경에 맞는 방법으로 설치/실행하자

지금까지가 Elastic Stack을 활용하기 위한 준비단계였다면,

이제는 본격적으로 Elastic Stack을 활용할 차례다.

담당자는 다음장의 workflow 대로 작업 할 수 있다.

Elastic Stack Workflow

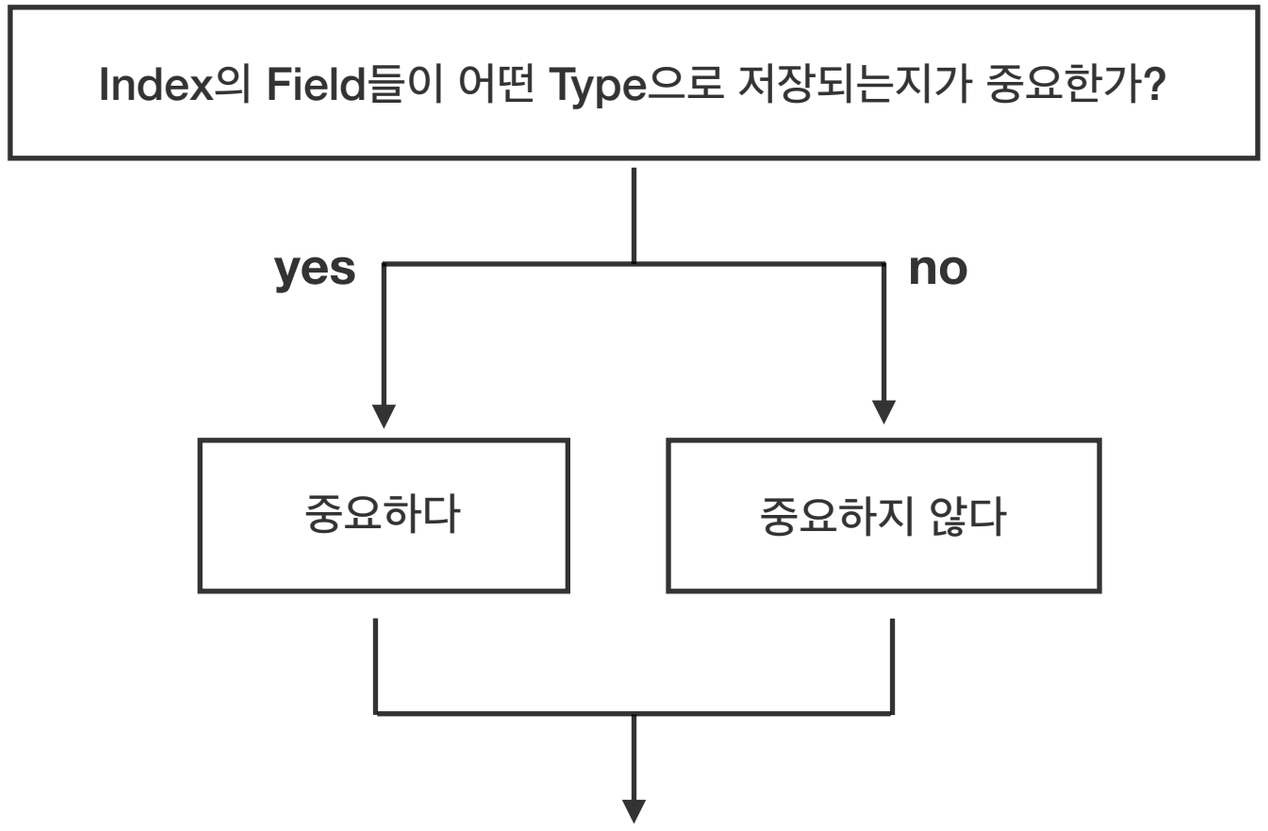
Elasticsearch	Logstash	Kibana
	Logstash Configuration 작성 및 실행	chart 선택 / aggregation 선택
Mapping 설정	데이터 수집 / 데이터 전처리 / 데이터 전송	데이터 등록 / (데이터 탐색) / 데이터 시각화 / 대시보드 제작

workflow

Mapping 설정 👑

1. Index 구성
   - 단발성 Index : nginx
   - 정기적 Index : nginx-2018.01.01, nginx-2018.01.02 ⇢ **template 활용 👑**
2. Document 구성
   - Document를 구성할 Field(s) 정하기
   - 예시 : nginx.access.response_code, nginx.access.geoip.location, …
3. Field Data Type 설정
   - 각 Field에 적절한 Data Type으로 설정
   - 예시
     - nginx.access.response_code : keyword
     - nginx.access.geoip.location : geo_point
     - nginx.access.body_sent.bytes : integer

Mapping 설정은 꼭 필요한가?



- Mapping이 없어도 에러가 발생하지는 않는다
- 다만 사용자가 원하는 Data Type으로 데이터가 저장된다는 보장이 없다. 예) “2018-01-01 13:00:00”
- 그러므로 (indexing 전에) 가급적 Mapping을 설정하는 걸 권장한다

Logstash Configuration 작성 및 실행

문제 정의

1. 데이터가 어디에 있는지 (=**input**) 명확히 한다.
2. 데이터를 어디로 전송할지 (=**output**) 명확히 한다.
3. 데이터를 어떻게 변형할지 (=**filter**) 명확히 한다.

코드 작성

4. 적절한 **input** plugin 선택
5. 적절한 **output** plugin 선택
6. 적절한 **filter** plugin 선택

Logstash Input Plugins 👑

plugin	input
stdin 👑	사용자 입력 메시지
file 👑	file의 각 line
jdbc 👑	database의 각 row (JDBC driver 이용)
elasticsearch 👑	elasticsearch index의 각 document

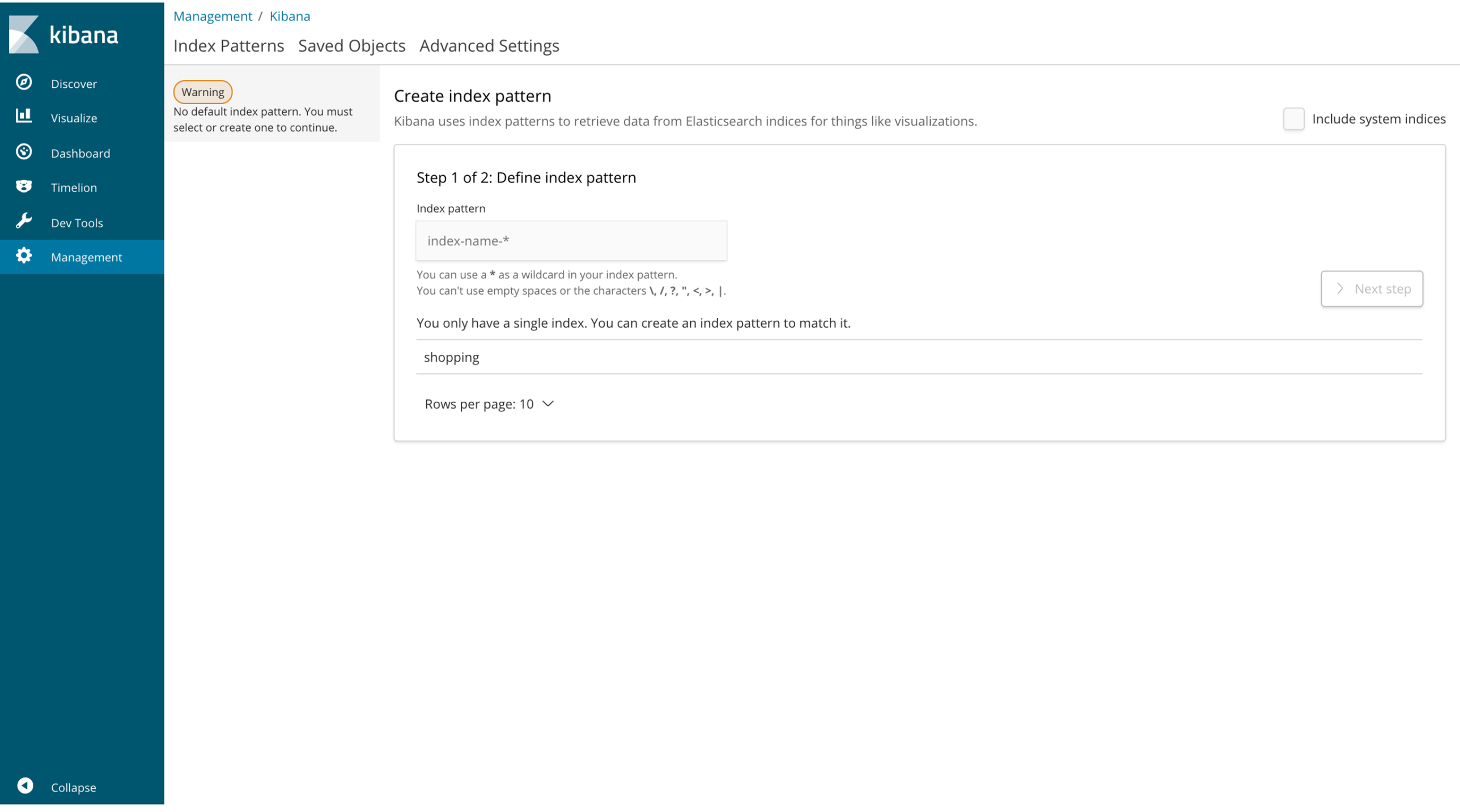
Logstash Output Plugins 👑

plugin	output
stdout 👑	console에 standard output 형태 출력
csv 👑	csv format 파일
elasticsearch 👑	elasticsearch

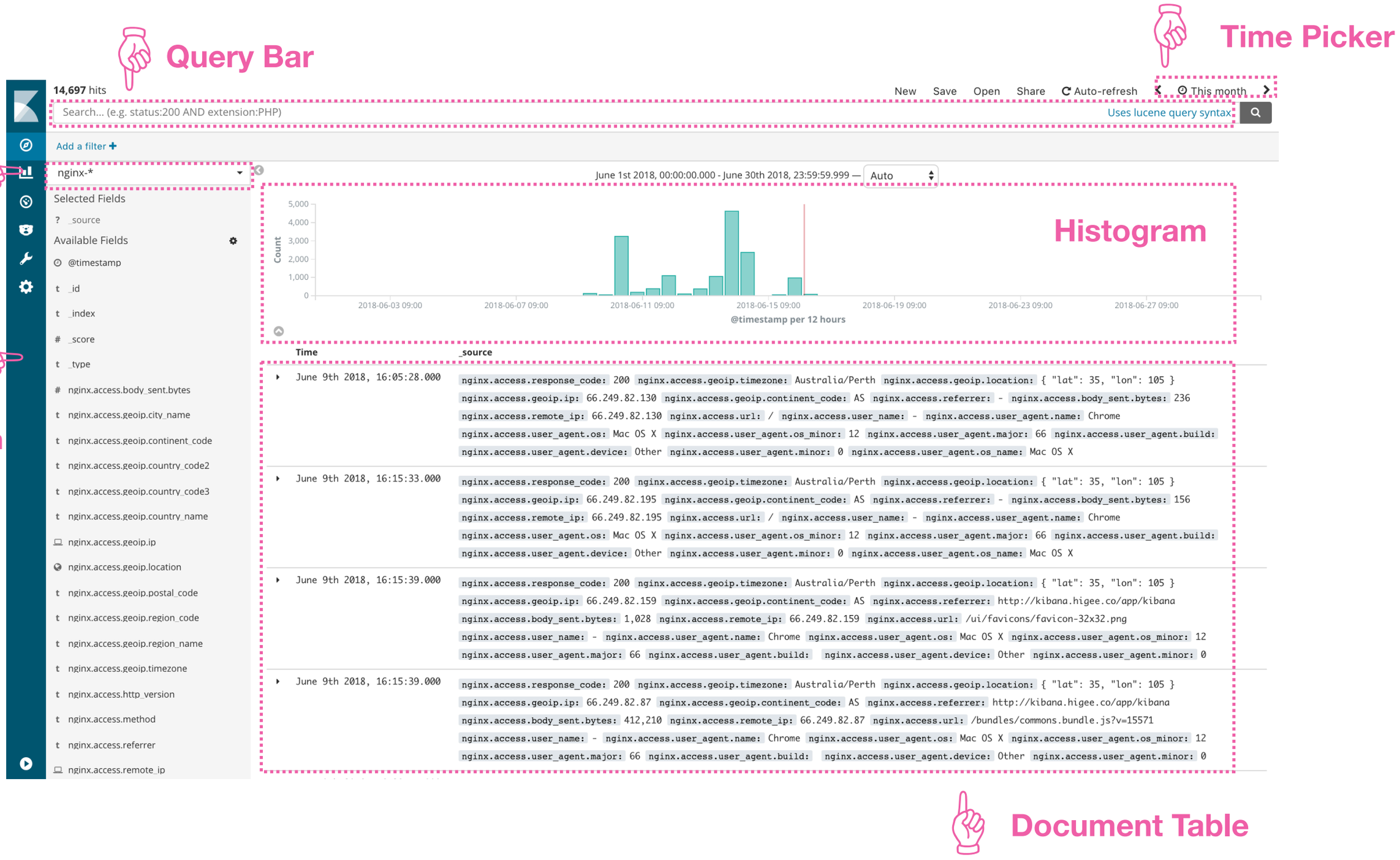
Logstash Filter Plugins ♕

plugin	역할
csv ♕	특정 field를 사용자 지정 separator로 나누기
mutate ♕	기본적인 데이터 변형 작업을 수행 (convert, rename, replace, add_field …)
date ♕	date field를 parse해서 timezone 변형 등의 date 관련 변형 작업 수행
drop ♕	특정 event cancel
grok ♕	access.log, apache.log 등 비정형 데이터를 parse 하는데 사용
ruby ♕	ruby code를 작성하여 custom filter 작성
elasticsearch ♕	event의 특정 field값으로 elasticsearch에 search/aggregation 수행 후 그 결과 반영
range ♕	message의 크기에 따라서 특정 field drop, tag 추가, field 추가 등의 작업 수행
truncate ♕	message의 길이가 기준치 이상일 경우 특정 길이를 넘지 않게 잘라낸다
⋮	⋮

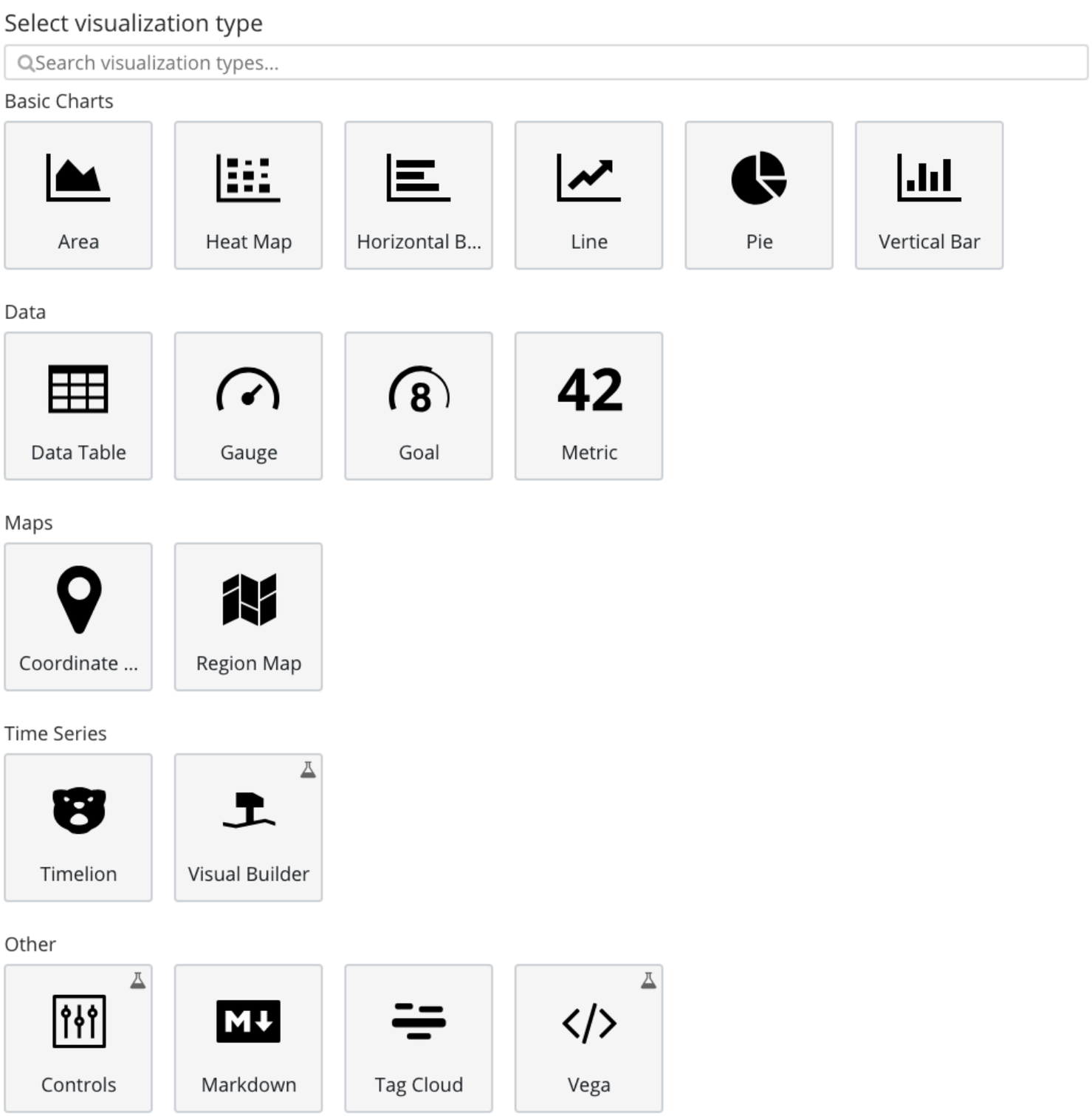
Index 등록 👑


kibana
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
Collapse
Management / Kibana
Index Patterns
Saved Objects
Advanced Settings
Warning
No default index pattern. You must select or create one to continue.
Create index pattern
Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.
Include system indices
Step 1 of 2: Define index pattern
Index pattern
index-name-*
You can use a * as a wildcard in your index pattern.
You can't use empty spaces or the characters \, /, ?, ", <, >, |.
Next step
You only have a single index. You can create an index pattern to match it.
shopping
Rows per page: 10

데이터 탐색 👑


Query Bar
Time Picker
Index Pattern
Side Navigation
Histogram
Document Table
14,697 hits
New
Save
Open
Share
Auto-refresh
This month
Search... (e.g. status:200 AND extension:PHP)
Uses lucene query syntax
Add a filter
nginx-*
June 1st 2018, 00:00:00.000 - June 30th 2018, 23:59:59.999 —
Auto
Selected Fields
_source
Available Fields
@timestamp
_id
_index
_score
_type
nginx.access.body_sent.bytes
nginx.access.geoip.city_name
nginx.access.geoip.continent_code
nginx.access.geoip.country_code2
nginx.access.geoip.country_code3
nginx.access.geoip.country_name
nginx.access.geoip.ip
nginx.access.geoip.location
nginx.access.geoip.postal_code
nginx.access.geoip.region_code
nginx.access.geoip.region_name
nginx.access.geoip.timezone
nginx.access.http_version
nginx.access.method
nginx.access.referrer
nginx.access.remote_ip
Count
5,000
4,000
3,000
2,000
1,000
0
2018-06-03 09:00
2018-06-07 09:00
2018-06-11 09:00
2018-06-15 09:00
2018-06-19 09:00
2018-06-23 09:00
2018-06-27 09:00
@timestamp per 12 hours
Time
_source
June 9th 2018, 16:05:28.000
nginx.access.response_code: 200 nginx.access.geoip.timezone: Australia/Perth nginx.access.geoip.location: { "lat": 35, "lon": 105 }
nginx.access.geoip.ip: 66.249.82.130 nginx.access.geoip.continent_code: AS nginx.access.referrer: - nginx.access.body_sent.bytes: 236
nginx.access.remote_ip: 66.249.82.130 nginx.access.url: / nginx.access.user_name: - nginx.access.user_agent.name: Chrome
nginx.access.user_agent.os: Mac OS X nginx.access.user_agent.os_minor: 12 nginx.access.user_agent.major: 66 nginx.access.user_agent.build:
nginx.access.user_agent.device: Other nginx.access.user_agent.minor: 0 nginx.access.user_agent.os_name: Mac OS X
June 9th 2018, 16:15:33.000
nginx.access.response_code: 200 nginx.access.geoip.timezone: Australia/Perth nginx.access.geoip.location: { "lat": 35, "lon": 105 }
nginx.access.geoip.ip: 66.249.82.195 nginx.access.geoip.continent_code: AS nginx.access.referrer: - nginx.access.body_sent.bytes: 156
nginx.access.remote_ip: 66.249.82.195 nginx.access.url: / nginx.access.user_name: - nginx.access.user_agent.name: Chrome
nginx.access.user_agent.os: Mac OS X nginx.access.user_agent.os_minor: 12 nginx.access.user_agent.major: 66 nginx.access.user_agent.build:
nginx.access.user_agent.device: Other nginx.access.user_agent.minor: 0 nginx.access.user_agent.os_name: Mac OS X
June 9th 2018, 16:15:39.000
nginx.access.response_code: 200 nginx.access.geoip.timezone: Australia/Perth nginx.access.geoip.location: { "lat": 35, "lon": 105 }
nginx.access.geoip.ip: 66.249.82.159 nginx.access.geoip.continent_code: AS nginx.access.referrer: http://kibana.higee.co/app/kibana
nginx.access.body_sent.bytes: 1,028 nginx.access.remote_ip: 66.249.82.159 nginx.access.url: /ui/favicons/favicon-32x32.png
nginx.access.user_name: - nginx.access.user_agent.name: Chrome nginx.access.user_agent.os: Mac OS X nginx.access.user_agent.os_minor: 12
nginx.access.user_agent.major: 66 nginx.access.user_agent.build: nginx.access.user_agent.device: Other nginx.access.user_agent.minor: 0
June 9th 2018, 16:15:39.000
nginx.access.response_code: 200 nginx.access.geoip.timezone: Australia/Perth nginx.access.geoip.location: { "lat": 35, "lon": 105 }
nginx.access.geoip.ip: 66.249.82.87 nginx.access.geoip.continent_code: AS nginx.access.referrer: http://kibana.higee.co/app/kibana
nginx.access.body_sent.bytes: 412,210 nginx.access.remote_ip: 66.249.82.87 nginx.access.url: /bundles/commons.bundle.js?v=15571
nginx.access.user_name: - nginx.access.user_agent.name: Chrome nginx.access.user_agent.os: Mac OS X nginx.access.user_agent.os_minor: 12
nginx.access.user_agent.major: 66 nginx.access.user_agent.build: nginx.access.user_agent.device: Other nginx.access.user_agent.minor: 0

Visualize - Chart 선택 👑



Visualize - Aggregation 선택 👑

1. 시각화 하려는 문제 정의
2. 문제에서 **metrics** 영역과 **buckets** 영역 구분
3. **metrics**와 **buckets** 내에서 사용할 aggregation 선택
4. term aggregation으로 **bucket**을 나눌 경우 sorting을 위한 aggregation 정의

Metric Aggregation 👑

종류	적용 가능 Type	상세
Avg 👑	Number	(Bucket 내) Document의 특정 Field의 평균 계산
Sum 👑	Number	(Bucket 내) Document의 특정 Field의 합 계산
Min/Max 👑	Number	(Bucket 내) Document의 특정 Field의 최소/최대 계산
Extended Stats 👑	Number	(Bucket 내) Document의 특정 Field의 기초 통계값 계산
Cardinality 👑	Number	(Bucket 내) Document의 특정 Field의 고유한 개수 계산
Percentiles 👑	Number	(Bucket 내) Document의 특정 Field의 백분위수 계산
Percentiles Ranks 👑	Number	(Bucket 내) Document의 특정 Field의 백분위 계산
Top Hits 👑	All	(Bucket 내) 특정 조건을 만족하는 Documents의 특정 Field의 Agg 반환
Value Count 👑	All	(Bucket 내) Document의 개수 계산

Bucket Aggregation 👑

종류	적용 가능 Type	상세
Date Histogram 👑	Date	날짜/시간을 일정하게 지정하여 구간 내의 Documents로 Bucket 형성
Date Range 👑	Date	날짜/시간을 임의로 지정하여 구간 내의 Documents로 Bucket 형성
Histogram 👑	Number	범위를 일정하게 지정하여 구간 내의 Documents로 Bucket 형성
Range 👑	Number	범위를 임의로 지정하여 구간 내의 Documents로 Bucket 형성
Terms 👑	Date, IP, Number, String	특정 Field 값을 기준으로 Bucket 생성
Significant Terms 👑	String	Background 대비 Foreground에서 특별한 Field 값으로 Bucket 생성
Filters 👑	Date, IP, Number, String	직접 조건을 입력하여 Bucket 생성
Geo Hash 👑	Geo Point	특정 지점 (Centroid) 근처에 있는 값들을 모아서 Bucket 생성
IPv4 Range 👑	IP	IP 주소의 범위로 Bucket 생성

Parent Pipeline Aggregation 👑

종류	설명
Derivative 👑	Date Histogram/Range Agg 및 Bucket 별 Metric Agg 후, 연속한 Bucket 간 차이를 구함
Cumulative Sum 👑	Date Histogram/Range Agg 및 Bucket 별 Metric Agg 후, Bucket들의 누적합을 구함
Moving Average 👑	Date Histogram/Range Agg 및 Bucket 별 Metric Agg 후, 연속한 {n개} Bucket 들의 평균을 구함
Serial Differencing 👑	Date Histogram/Range Agg 및 Bucket 별 Metric Agg 후, {n번째 이전} Bucket 과의 차이를 구함

Sibling Pipeline Aggregation 👑

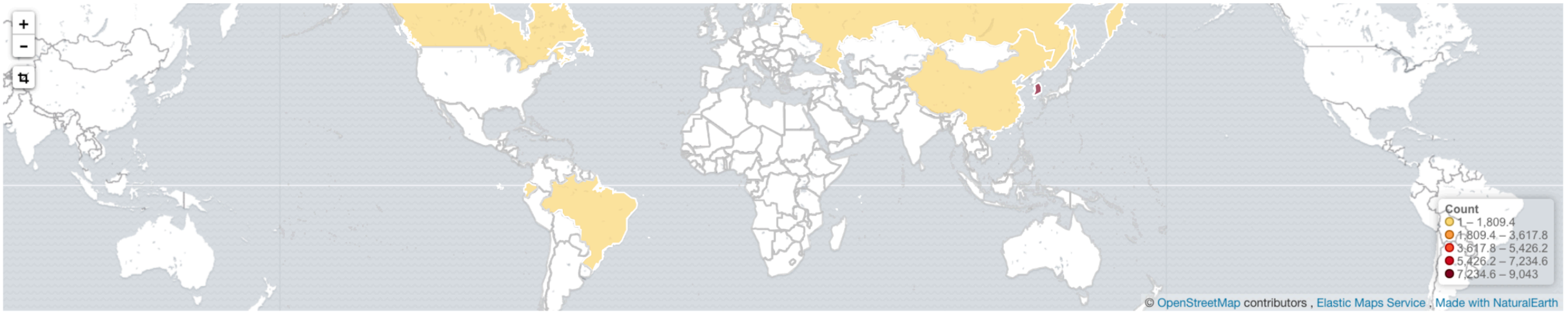
종류	설명
Min Bucket 👑	Bucket Agg 및 Bucket 별 Metric Agg 후, Min Aggregation 적용
Max Bucket 👑	Bucket Agg 및 Bucket 별 Metric Agg 후, Max Aggregation 적용
Sum Bucket 👑	Bucket Agg 및 Bucket 별 Metric Agg 후, Sum Aggregation 적용
Average Bucket 👑	Bucket Agg 및 Bucket 별 Metric Agg 후, Average Aggregation 적용

Dashboard 👑

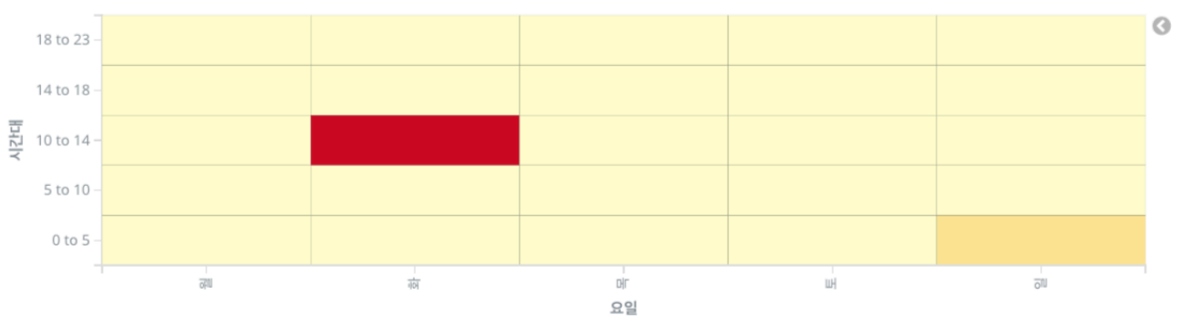
Nginx Access.Log Dashboard

```
66.249.82.131 - - [13/Jun/2018:17:01:02 +0000] "GET /ui/favicons/favicon-16x16.png HTTP/1.1" 304 0 "http://kibana.higee.co/app/kibana" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like G
118.221.38.242 - - [13/Jun/2018:17:01:14 +0000] "POST /api/console/proxy?path=_aliases&method=GET HTTP/1.1" 200 107 "http://kibana.higee.co/app/kibana" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.
118.221.38.242 - - [13/Jun/2018:17:01:14 +0000] "POST /api/console/proxy?path=_mapping&method=GET HTTP/1.1" 200 974 "http://kibana.higee.co/app/kibana" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.
66.249.82.131 - - [13/Jun/2018:17:01:16 +0000] "GET /ui/favicons/favicon-32x32.png HTTP/1.1" 304 0 "http://kibana.higee.co/app/kibana" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like G
66.249.82.129 - - [13/Jun/2018:17:01:17 +0000] "GET /ui/favicons/favicon-16x16.png HTTP/1.1" 304 0 "http://kibana.higee.co/app/kibana" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like G
```

[nginx] region maps



[nginx] heat map



[nginx] tag-cloud

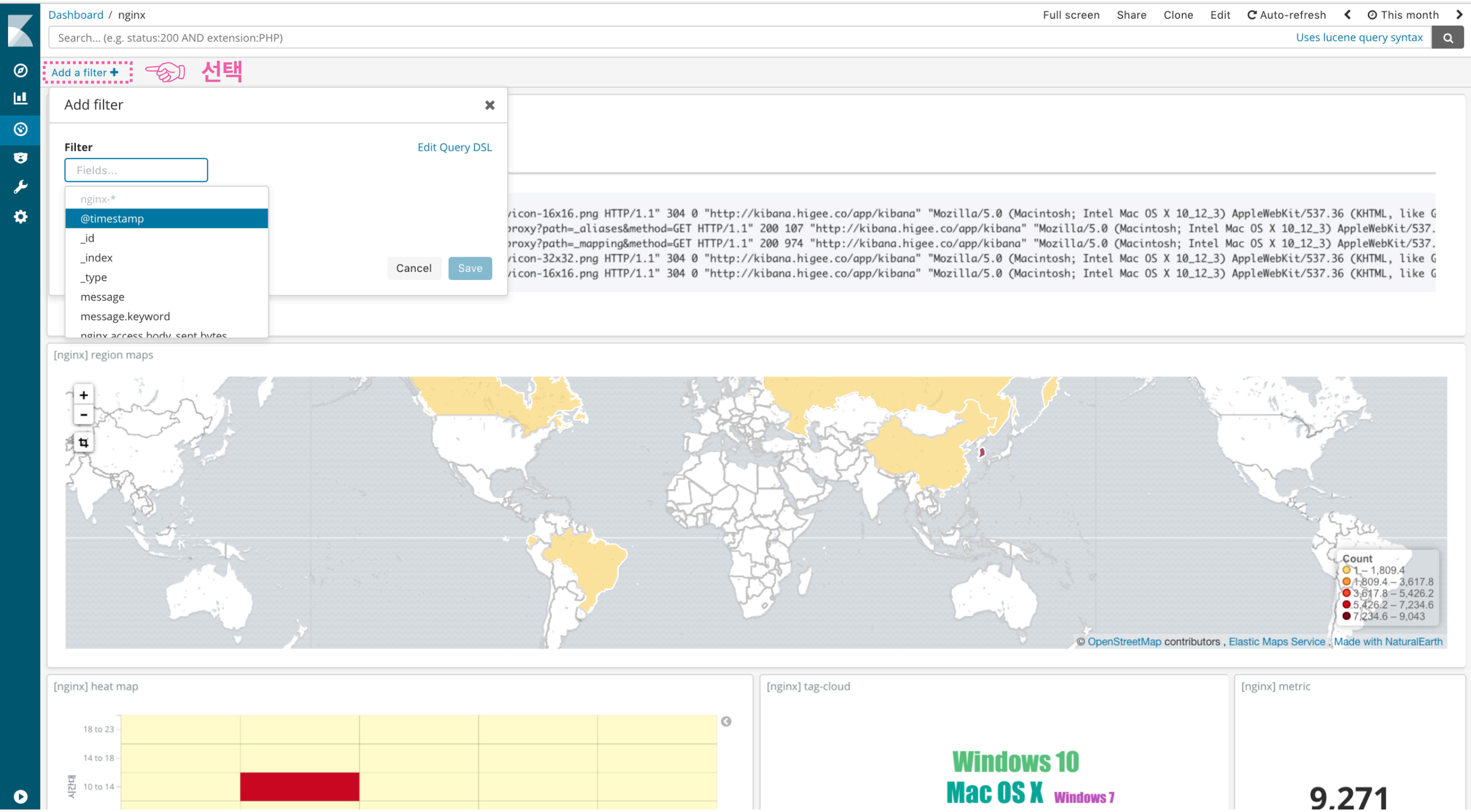
[nginx] metric

9,271
req

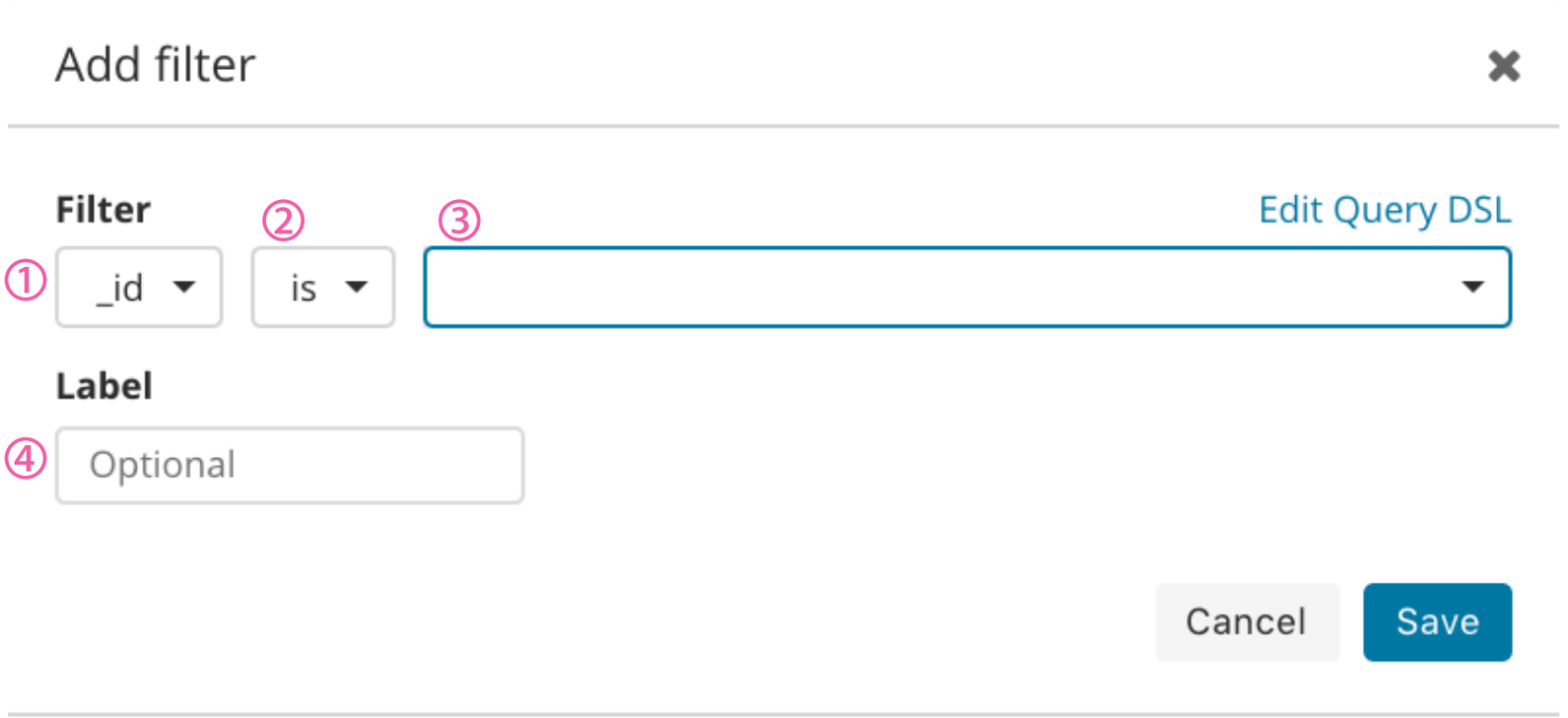
Dashboard는 만들었는데

원하는 조건의 데이터만 보고 싶으면?

Filter를 실행하자 ♛



Filter 사용법을 익히자 👑



① Filter 적용할 Field 선택

② 적용할 Operator 선택 (다음 페이지 참조)

③ Filter에 적용하려는 Value 입력

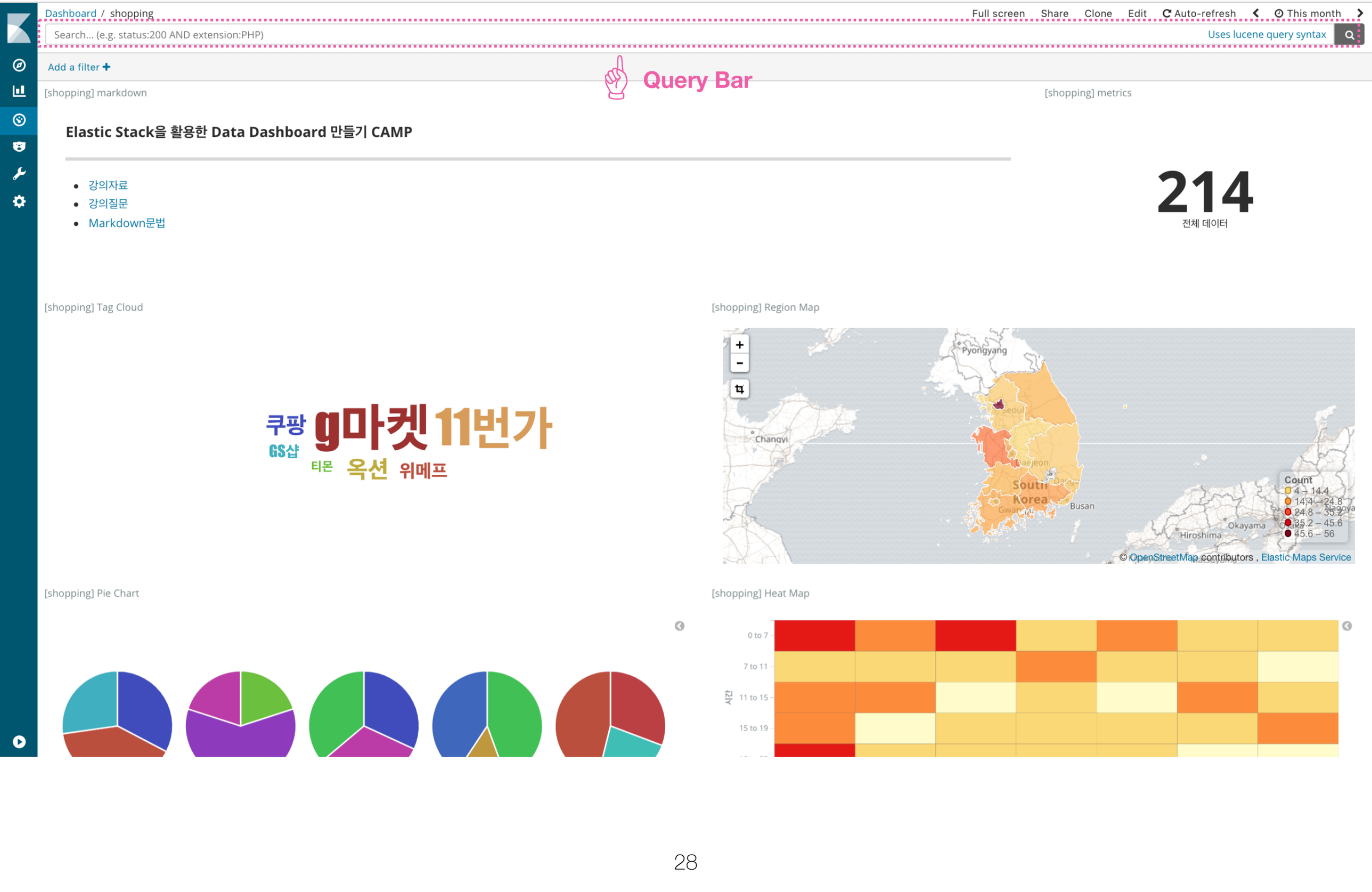
④ (여러 Filter 구분하기 위한) 이름 입력

Operator 설명 👑

Operator	역할
is	Field의 Value가 입력한 값과 일치하는 Documents 선택
is not	Field의 Value가 입력한 값과 일치하지 않는 Documents 선택
is one of	Field의 Value가 입력한 값 중에 존재하는 Documents 선택
is not one of	Field의 Value가 입력한 값 중에 존재하지 않는 Documents 선택
exists	Field가 적어도 한 개의 non-null 값을 가지는 Documents 선택
does not exist	Field가 존재하지 않거나 null 값만 가지는 Documents 선택
is between	Field의 Value가 입력한 값 사이에 존재하는 Documents 검색
is not between	Field의 Value가 입력한 값 사이에 존재하지 않는 Documents 검색

구글 검색처럼 검색할 수는 없나?

Query Bar를 확인하자



Lucene Query의 사용법을 익히자 👑

종류	기능	예시
Keyword 검색	Field에 상관없이 검색어와 일치하는 Doc 검색	여성
Field Match 검색	특정 Field 값이 검색어와 일치하는 Doc 검색	고객성별:여성
Exact Match 검색	특정 Field 값이 검색어와 정확히 일치하는 Doc 검색	배송메모: "상품 이상"
Exists 검색	특정 Field가 non-null value를 가진 Doc 검색	_exists_:구매사이트
Term 검색	특정 Field 값이 검색어 중 하나라도 일치하는 Doc 검색	상품분류: ("니트" "코트")
Fuzzy 검색	검색어와 유사한 Doc 검색	경상복도~
Proximity 검색	검색어의 순서를 변경해서 Doc 검색	배송메모: "내에 시간 배송 못함"~2
Numeric Value 검색	특정 Field 값이 입력값보다 큰 (또는 작은) Doc 검색	상품가격:>5000
Range 검색	특정 Field 값이 입력값 사이에 있는 Doc 검색	고객나이: [10 TO 30}
Wildcard ? 검색	Wildcard ? (한글자)를 활용해서 Doc 검색	서?특별시
Wildcard * 검색	Wildcard * (생략 혹은 그 이상)를 활용해서 Doc 검색	상품*:셔츠
OR 연산 (||)	여러 검색 조건들을 OR로 묶어 검색 수행	고객성별: 여성 OR 상품분류:셔츠
AND 연산 (&&)	여러 검색 조건들을 AND로 묶어 검색 수행	고객성별: 여성 AND 상품분류:셔츠
NOT 연산 (!)	뒤이어 오는 조건을 부정해서 검색 수행	NOT 구매사이트:옥션
Must be Present 연산	바로 뒤에 오는 조건을 만족하는 Doc 검색	+예약여부:예약
Must not be Present 연산	바로 뒤에 오는 조건을 만족하지 않는 Doc 검색	-구매사이트:11번가

Filter는 편하지만 기능이 제한적이고,

Search는 Scripted Field가 검색이 안된다.

두 개를 아우르고 싶다면?

Query DSL로 무얼 할 수 있을까? 👑

Match All Query	Full Text Queries	Term Level Queries	Specialized Queries	Compound Queries
match-all	match query-string ⋮	exists fuzzy prefix range term terms wildcard ⋮	script ⋮	bool ⋮

Bool Query로 여러가지 Query를 함께 사용할 수 있다 👑

A : 고객주소_시도 = 서울특별시 — **Term Query**

B : 구매사이트 = 11로 시작 — **Prefix Query**

C : 고객나이 < 30 — **Range Query**

D : 주문시간 > 15 — **Script Query**

↓ **검색 가능**

- A AND B
- A AND NOT B
- A OR B
- A AND (B OR C)
- A AND (B OR C OR D 중 2개 이상 만족)

⋮

```
GET /shopping/_search
{
  "query": {
    "bool": {
      "must": [
        {"term": {"고객주소_시도": "서울특별시"}}
      ],
      “must_not” : [
        {"prefix": {"구매사이트" : "11"}}
      ],
      "should": [
      {"range": {"고객나이": { "lt": 30}}},
       {"script": {
          "script" : {
            "source" : "Instant.ofEpochMilli(doc[‘주문시간'].value.millis).atZone(ZoneId.of('Asia/Seoul')).hour>15"}}}
      ],
      "minimum_should_match": 2
    }
  }
}
```

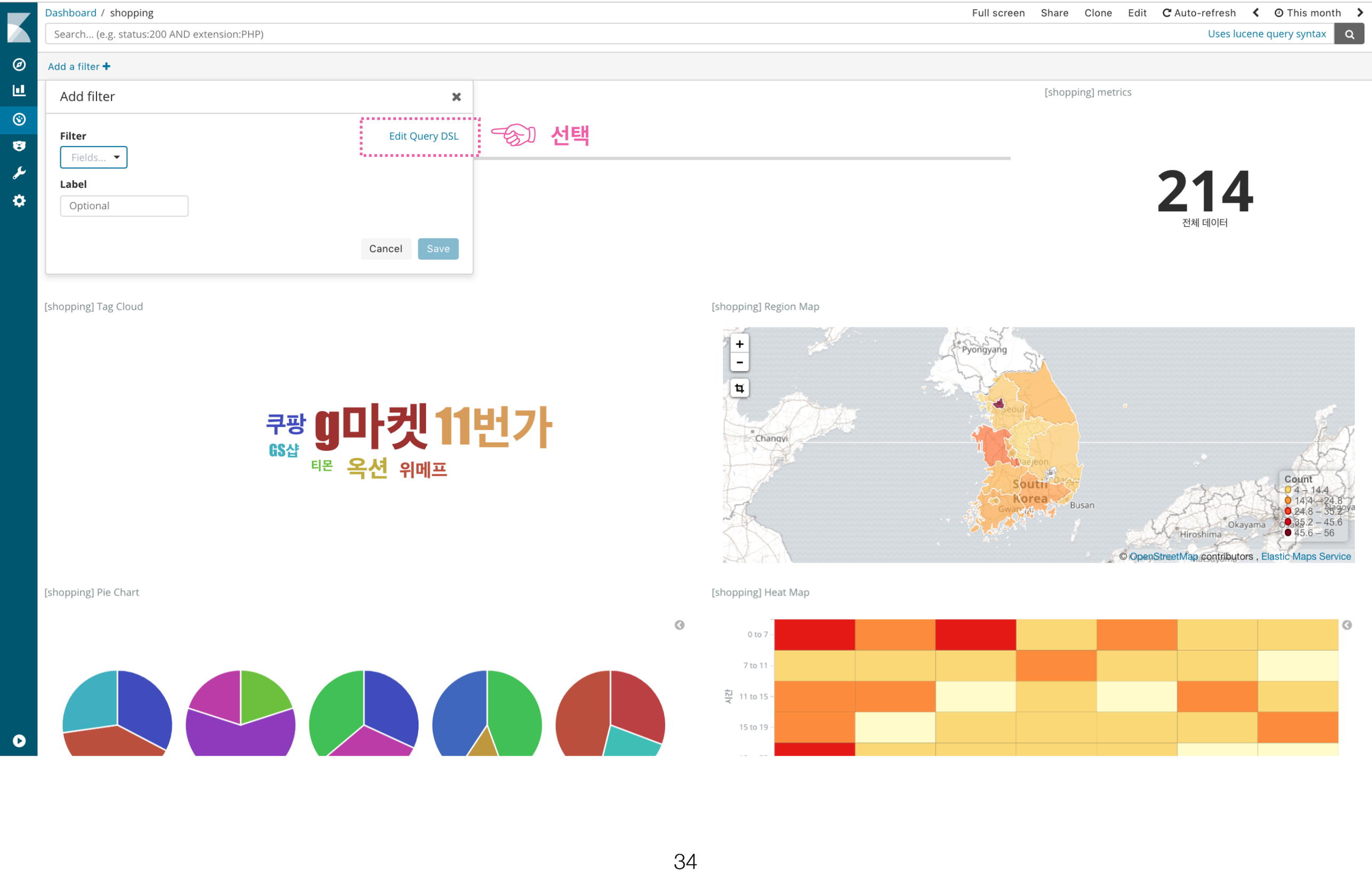
☞ 반드시 만족해야 한다

☞ 반드시 만족하면 안된다

☞ {minimum_should_match}개 이상 만족해야 한다

☞ should clause 내의 query가 n개 이상 참이어야 한다

Filter에 Query DSL을 적용하자 1



Filter에 Query DSL을 적용하자 2

Dashboard / shopping

Full screen Share Clone Edit Auto-refresh This month

Search... (e.g. status:200 AND extension:PHP) Uses lucene query syntax

Add a filter +

Add filter

Query DSL 입력

Filter Search filter values

```
{
  "query": {
    "bool": {
      "must": [
        {"term": {"고객주소_시도": "서울특별시"}}
      ],
      "must_not" : [
        {"prefix": {"구매사이트" : "11"}}
      ],
      "should": [
```

Filters are built using the Elasticsearch Query DSL.

Label

테스트

Cancel Save

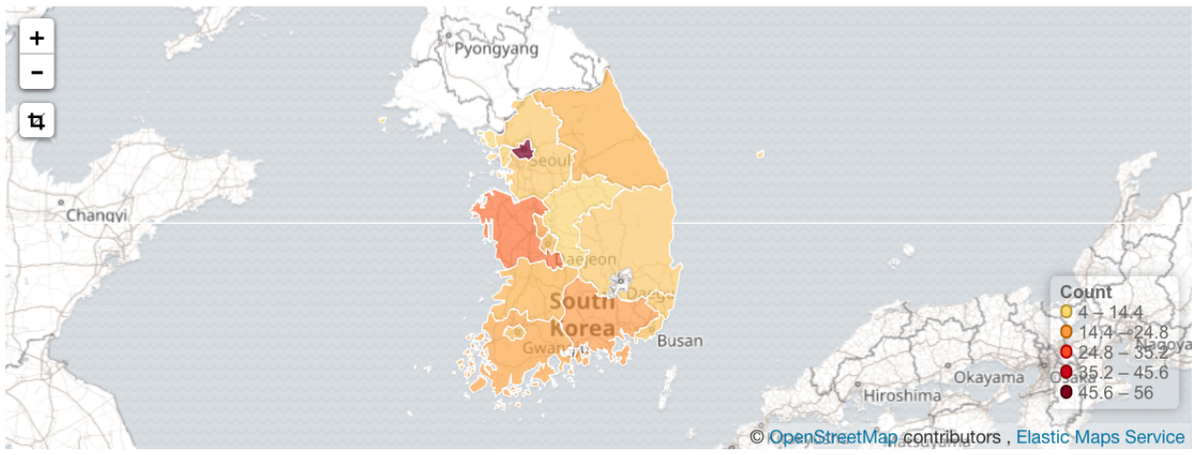
쿠팡 g마켓 11번가 GS샵 티몬 옥션 위메프

[shopping] metrics

214

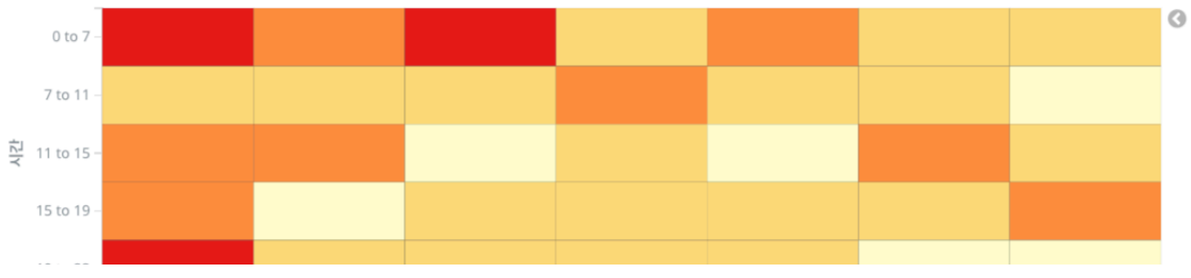
전체 데이터

[shopping] Region Map

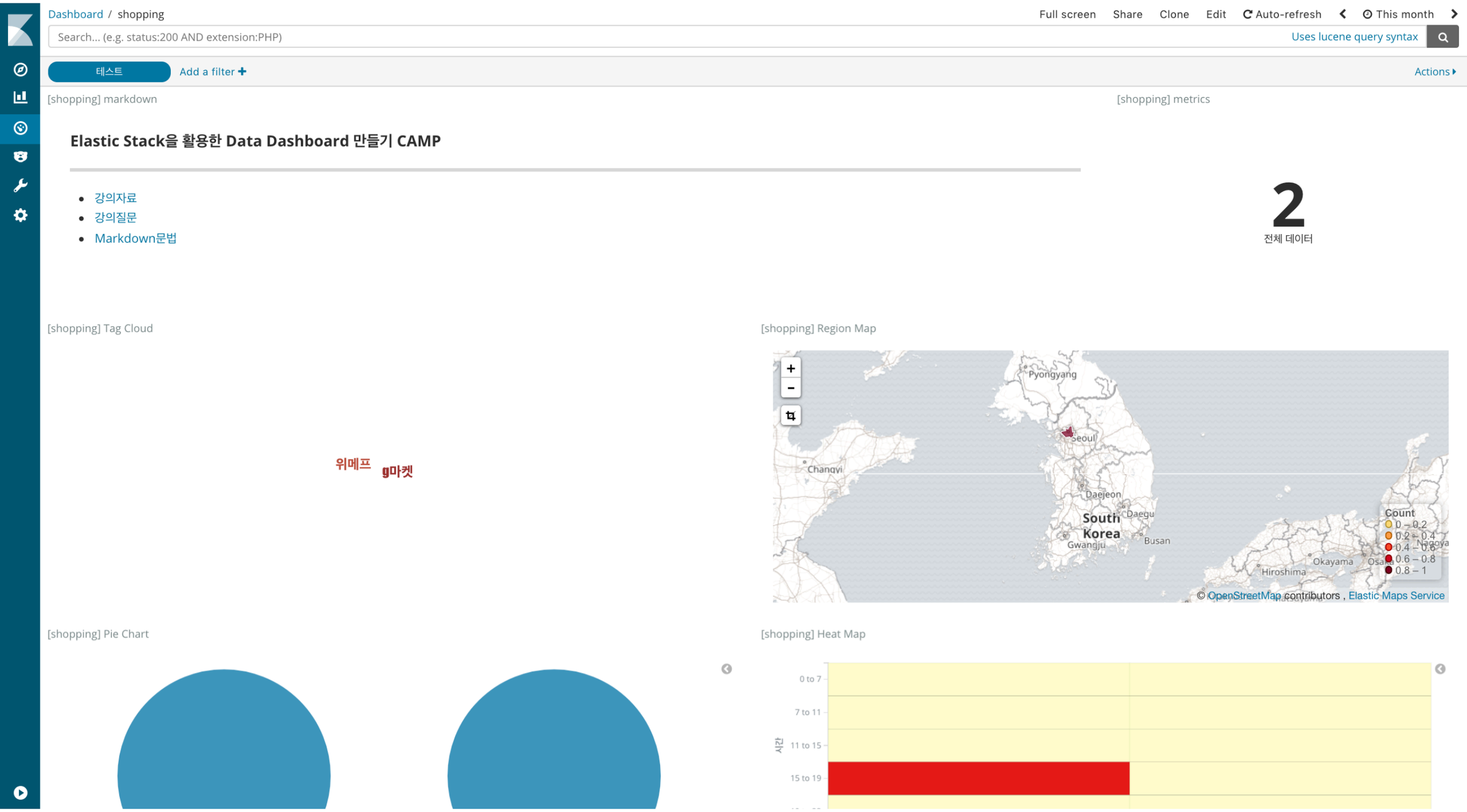


[shopping] Pie Chart

[shopping] Heat Map



Filter에 Query DSL을 적용한 결과를 보자



2. 최종 실습

안내

- 실제로 대시보드 구축 프로젝트를 가정하고 실습 진행
- 자세한 안내보다는 **요구사항 위주의 문서**
- 최소한의 정보는 제공하지만 **직접 데이터를 조회하면서 작업**해야 함
- **시각화의 경우 정해진 답이 없으므로**, 사용자의 니즈를 충족하는 범위 내에서 **자유롭게** 제작

데이터 소개 - Chicago Divvy bicycle sharing system 👑

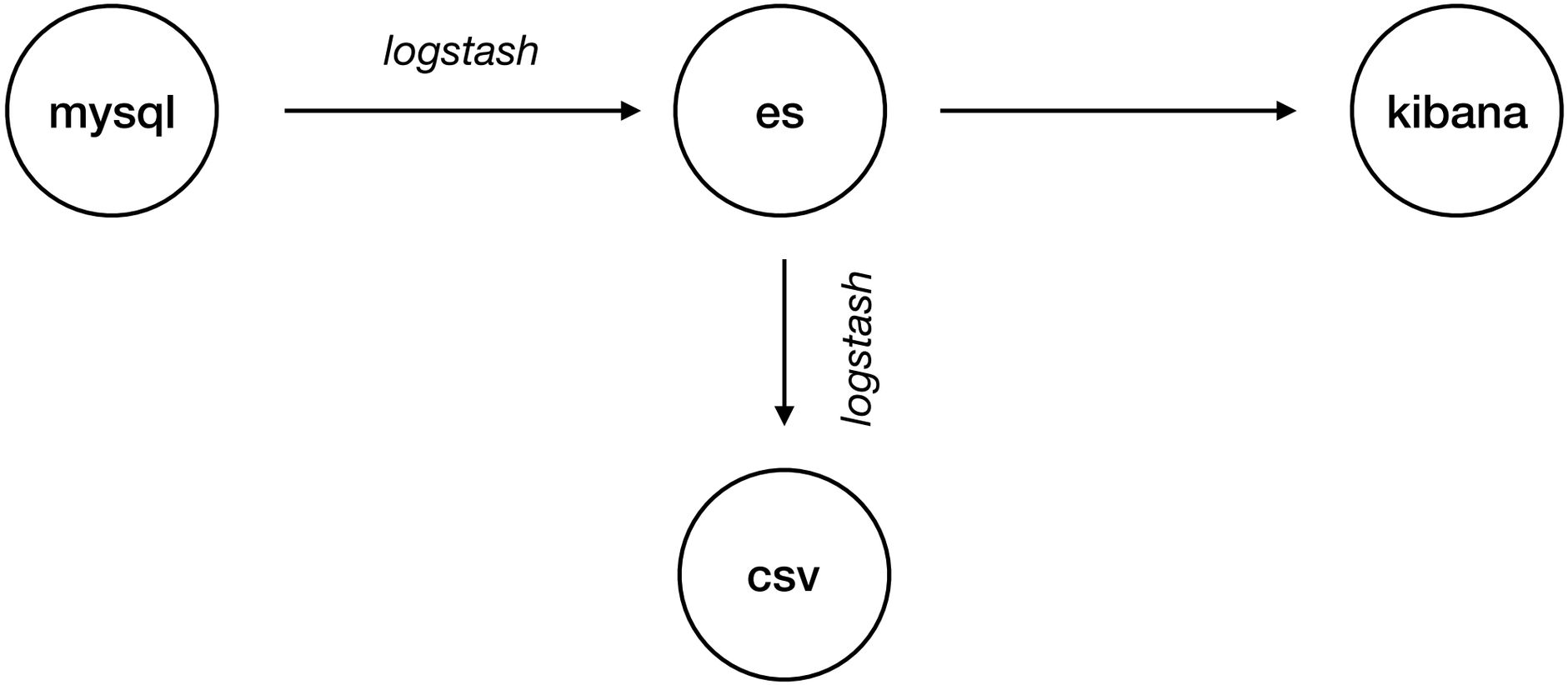
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	trip_id	year	month	week	day	hour	usertype	gender	starttime	stoptime	tripduration	temperature	events	from_station_id	from_station_name	latitude_start	longitude_start	dpcapacity_start	to_station_id	to_station_name	latitude_end	longitude_end	dpcapacity_end
2	16805064	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:44	3.63333333	60.1	cloudy	282	Halsted St & Maxwell St	41.864883	-87.647071	15	108	Halsted St & Polk St	41.87184	-87.64664	19
3	16805063	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:56	15.8	60.1	cloudy	146	Loomis St & Jackson Blvd	41.877945	-87.662007	11	125	Rush St & Hubbard St	41.890173	-87.626185	23
4	16805062	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 8:08	27.8	60.1	cloudy	230	Lincoln Ave & Roscoe St	41.94334	-87.67097	19	18	Wacker Dr & Washington St	41.883132	-87.637321	19
5	16805060	2017	10	40	3	7	Subscriber	Female	05/10/2017 7:40	05/10/2017 7:54	14.0166667	60.1	cloudy	264	Stetson Ave & South Water St	41.886835	-87.62232	19	53	LaSalle (Wells) St & Huron St	41.894882	-87.632326	27
6	16805059	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:48	7.21666667	60.1	cloudy	329	Lake Shore Dr & Diversey Pkwy	41.932684	-87.63625	15	144	Larrabee St & Webster Ave	41.921822	-87.64414	19
7	16805058	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:48	7.85	60.1	cloudy	217	Racine Ave (May St) & Fulton St	41.886926	-87.656919	15	18	Wacker Dr & Washington St	41.883132	-87.637321	19
8	16805057	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:49	9.03333333	60.1	cloudy	75	Canal St & Jackson Blvd	41.877245	-87.639366	31	44	State St & Randolph St	41.8847302	-87.62773357	27
9	16805056	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:47	6.33333333	60.1	cloudy	479	Drake Ave & Montrose Ave	41.9611541	-87.71656907	15	489	Drake Ave & Addison St	41.947326	-87.717582	15
10	16805055	2017	10	40	3	7	Subscriber	Female	05/10/2017 7:40	05/10/2017 7:43	2.98333333	60.1	cloudy	289	Wells St & Concord Ln	41.912133	-87.634656	19	289	Wells St & Concord Ln	41.912133	-87.634656	19
11	16805054	2017	10	40	3	7	Subscriber	Female	05/10/2017 7:40	05/10/2017 7:52	11.4166667	60.1	cloudy	119	Ashland Ave & Lake St	41.88541	-87.66732	19	164	Franklin St & Lake St	41.885837	-87.6355	27
12	16805053	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:46	5.46666667	60.1	cloudy	394	Clark St & 9th St (AMLI)	41.870816	-87.631246	15	287	Franklin St & Monroe St	41.880317	-87.635185	27
13	16805051	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:57	16.5666667	60.1	cloudy	291	Wells St & Evergreen Ave	41.906724	-87.63483	19	40	LaSalle St & Adams St	41.8793444	-87.63198522	15
14	16805050	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:48	8.45	60.1	cloudy	198	Green St & Madison St	41.881892	-87.648789	19	100	Orleans St & Merchandise Mart Plaza	41.888243	-87.63639	35
15	16805049	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 8:07	27.2833333	60.1	cloudy	338	Calumet Ave & 18th St	41.857611	-87.619407	15	328	Ellis Ave & 58th St	41.788746	-87.601334	19
16	16805048	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:48	8.41666667	60.1	cloudy	31	Franklin St & Chicago Ave	41.896776	-87.635633	23	286	Franklin St & Quincy St	41.878724	-87.634793	23
17	16805047	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:42	2.5	60.1	cloudy	38	Clark St & Lake St	41.8860208	-87.63087606	27	194	Wabash Ave & Wacker Pl	41.886875	-87.62603	19
18	16805046	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:44	4.03333333	60.1	cloudy	38	Clark St & Lake St	41.8860208	-87.63087606	27	125	Rush St & Hubbard St	41.890173	-87.626185	23
19	16805044	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:48	8.4	60.1	cloudy	255	Indiana Ave & Roosevelt Rd	41.867888	-87.623041	39	37	Dearborn St & Adams St	41.8793564	-87.62979104	19
20	16805043	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:53	13.4	60.1	cloudy	394	Clark St & 9th St (AMLI)	41.870816	-87.631246	15	51	Clark St & Randolph St	41.8845762	-87.63188991	39
21	16805042	2017	10	40	3	7	Subscriber	Female	05/10/2017 7:40	05/10/2017 7:47	7.08333333	60.1	cloudy	596	Benson Ave & Church St	42.048214	-87.683485	15	605	University Library (NU)	42.052939	-87.673447	15
22	16805041	2017	10	40	3	7	Subscriber	Female	05/10/2017 7:40	05/10/2017 8:06	26.1	60.1	cloudy	207	Emerald Ave & 28th St	41.84358	-87.645368	15	51	Clark St & Randolph St	41.8845762	-87.63188991	39
23	16805040	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 8:00	20.2833333	60.1	cloudy	87	Racine Ave & Fullerton Ave	41.9255626	-87.65840426	19	66	Clinton St & Lake St	41.885637	-87.641823	23
24	16805039	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:55	15.0333333	60.1	cloudy	163	Damen Ave & Clybourn Ave	41.931931	-87.677856	15	60	Dayton St & North Ave	41.910578	-87.64942193	19
25	16805038	2017	10	40	3	7	Subscriber	Female	05/10/2017 7:40	05/10/2017 7:54	14.9333333	60.1	cloudy	215	Damen Ave & Madison St	41.88137	-87.67493	15	241	Morgan St & Polk St	41.871737	-87.65103	19
26	16805037	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:42	2.4	60.1	cloudy	300	Broadway & Barry Ave	41.937725	-87.644095	19	156	Clark St & Wellington Ave	41.9364968	-87.64753866	15
27	16805036	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:40	05/10/2017 7:51	11.0833333	60.1	cloudy	306	Sheridan Rd & Buena Ave	41.958494	-87.654966	19	344	Ravenswood Ave & Lawrence Ave	41.96909	-87.674237	39
28	16805033	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:39	05/10/2017 7:48	8.91666667	60.1	cloudy	331	Halsted St & Blackhawk St (*)	41.908537	-87.648627	20	69	Damen Ave & Pierce Ave	41.909396	-87.67769193	19
29	16805032	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:39	05/10/2017 7:51	11.1	60.1	cloudy	74	Kingsbury St & Erie St	41.893882	-87.641711	23	283	LaSalle St & Jackson Blvd	41.87817	-87.631985	35
30	16805031	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:39	05/10/2017 7:54	14.15	60.1	cloudy	214	Damen Ave & Grand Ave	41.89122	-87.67686	23	38	Clark St & Lake St	41.8860208	-87.63087606	27
31	16805030	2017	10	40	3	7	Subscriber	Male	05/10/2017 7:39	05/10/2017 7:45	5.28333333	60.1	cloudy	19	Loomis St & Taylor St (*)	41.869417	-87.660996	16	342	Wolcott Ave & Polk St	41.871262	-87.673688	15

9,495,236 개

개요

- mysql에서 적당한 변형을 거쳐 elasticsearch로 데이터 전송
- kibana를 통해 대시보드 구축
- Filter, Query DSL, Lucene Query Syntax 등을 이용해 원하는 정보 검색
- Elasticsearch에서 특정 조건을 만족하는 Documents만 선별하여 csv 추출

개요



AWS EC2 Instance에 접속하자 👑

Elastic Stack (6.2.4) 설치/실행하자 👑

~~(시간 상 생략하고 기존 docker 사용)~~

template 생성 👑

- index pattern : chicago-bike-*
- type : chicago-bike

template mapping 설정 👑

Field	Data Type
starttime	date
stoptime	date
idx	integer
year	integer
month	integer
week	integer
day	integer
hour	integer
dpcapacity_start	integer
dpcapacity_end	integer
start_gps	geo_point
end_gps	geo_point
from_station_name	keyword
from_station_id	keyword
to_station_name	keyword
to_station_id	keyword
events	keyword
gender	keyword
usertype	keyword
tags	keyword
temperature	float
tripduration	float

logstash input plugin 👑

기본 정보

- input plugin : jdbc (mysql)
- host : 52.78.134.20:3306
- database : fc
- user/password : fc/week6
- table : chicago_bike
- query : SELECT * FROM chicago_bike;

주의

- jdbc_connection_string => ”jdbc:mysql://52.78.134.20:3306/fc?**useCursorFetch=true**”
- jdbc_fetch_size => 1000

logstash output plugin 👑

- output plugin : elasticsearch
- host : 각자 서버에 떠있는 elasticsearch
- index : chicago-bike-{year}.{month}
- type : chicago-bike
- id : {trip_id} 값으로 dynamic 하게 설정

예시) year:2014, month:10 trip_id:123인 경우

- index : chicago-bike-2014.10
- id : 123

logstash filter plugin 1 - conditional 👑, drop 👑, mutate 👑

- trip_duration ≤ 3 : drop an event
- trip_duration ≥ 24 : add tag “VIP”

logstash filter plugin 2 - mutate 👑

- “start_gps” field 생성 : latitude_start, longitude_start 조합
- “end_gps” field 생성 : latitude_end, longitude_end 조합
- 필드 제거 : @timestamp, @version, longitude_start, latitude_start, longitude_end, latitude_end

예시

latitude_start	-87.646697
longitude_start	41.8538017
latitude_end	-87.6614990
longitude_end	41.85760116

→

start_gps	-87.646697, 41.8538017
end_gps	-87.6614990, 41.85760116

logstash filter plugin 3 - date 👑

starttime과 stoptime field의 timezone 변환

- format : YYYY-MM-dd HH:mm:ss
- timezone : America/Chicago

Kibana에서 Index Patterns를 등록하자 👑

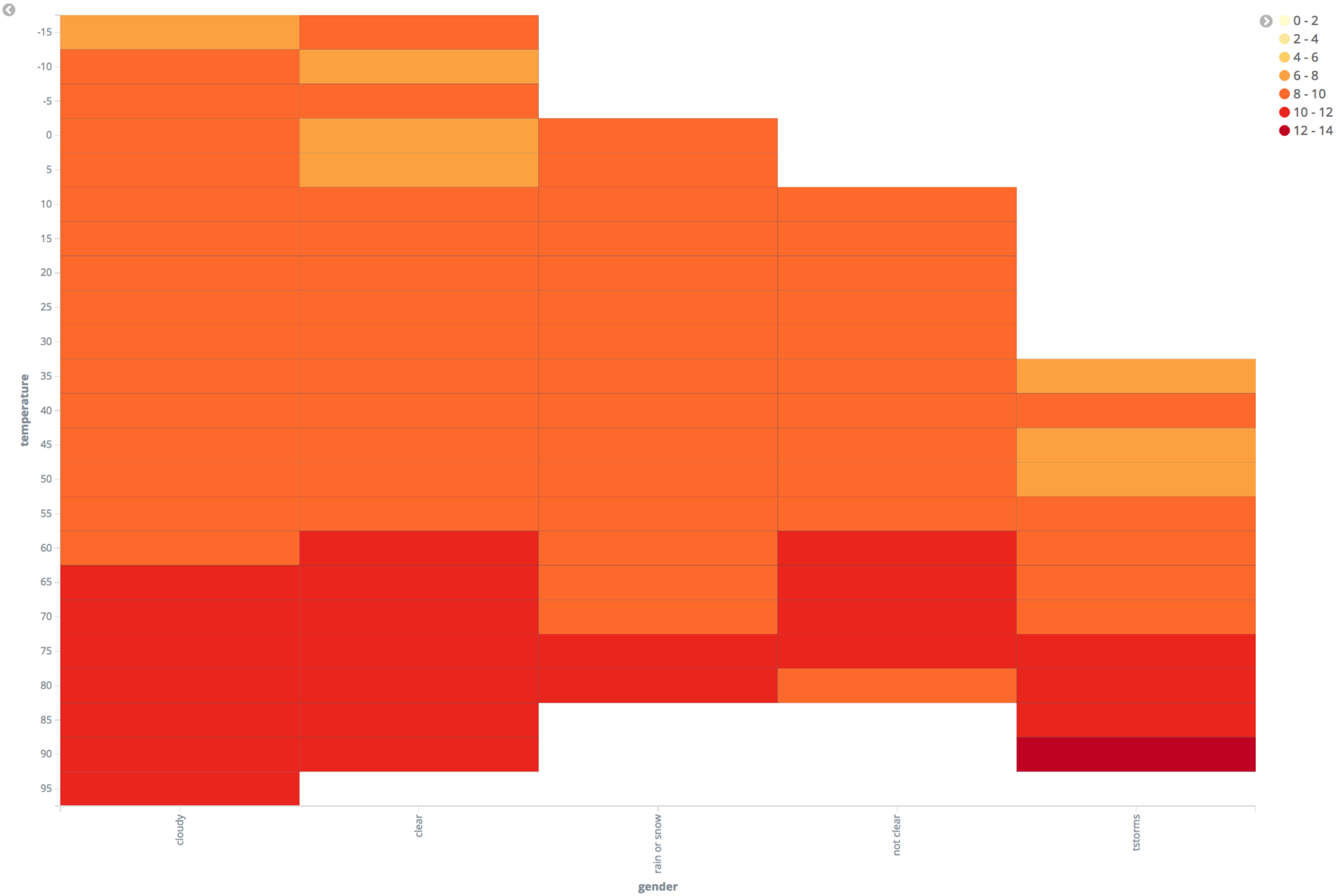
- Index Pattern : chicago-bike-*
- time filter field : starttime
- time range : 2014년 1월 1일 ~ 2017년 12월 31일

Discover에서 다음과 같은 질문에 답해보자 👑

- 전체 Documents 개수는?
- tripduration의 최대값은?
- hour가 13~17시 사이인 Documents의 개수는?
- “VIP” tag가 달린 Documents의 개수는?
- 일별로 봤을 때 가장 Documents가 많았던 일은?
- tripduration로 내림차순 정렬했을 때 상위 500개 Documents의 gender field 비율은?

Visualize - Heat Map 👑

- Buckets
  - x 축 : events field를 기준으로 documents count가 많았던 상위 5개 events 값 선정
  - y 축 : temperature field를 기준으로 interval이 5인 histogram aggregation 적용
- Metrics : tripduration field의 중위값


0 - 2
2 - 4
4 - 6
6 - 8
8 - 10
10 - 12
12 - 14
temperature
-15
-10
-5
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
cloudy
clear
rain or snow
not clear
tstorms
gender

Visualize - Coordinate Map 👑

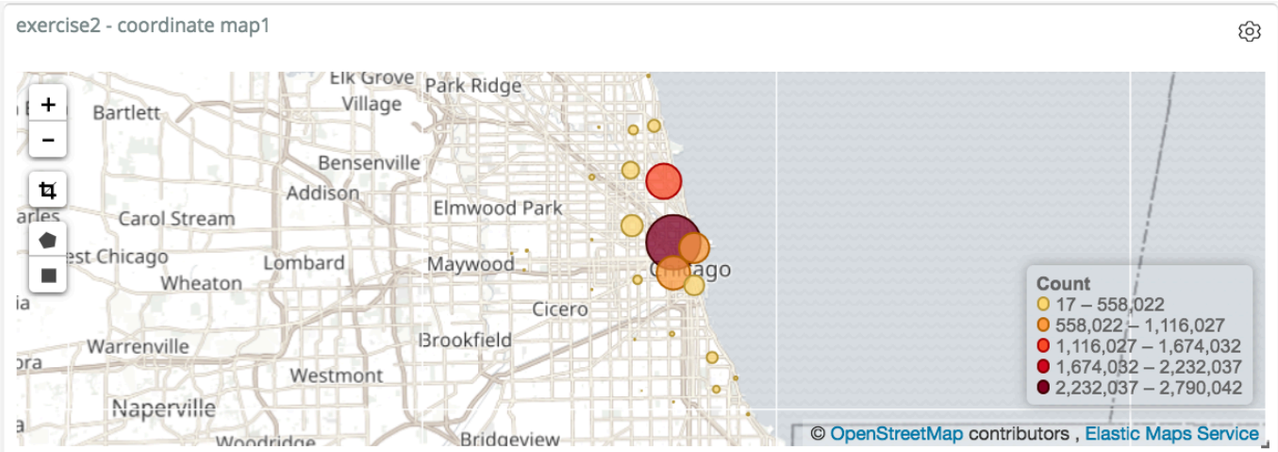
- Buckets : start_gps field를 기준으로 Geohash aggregation
- Metrics : Documents Count

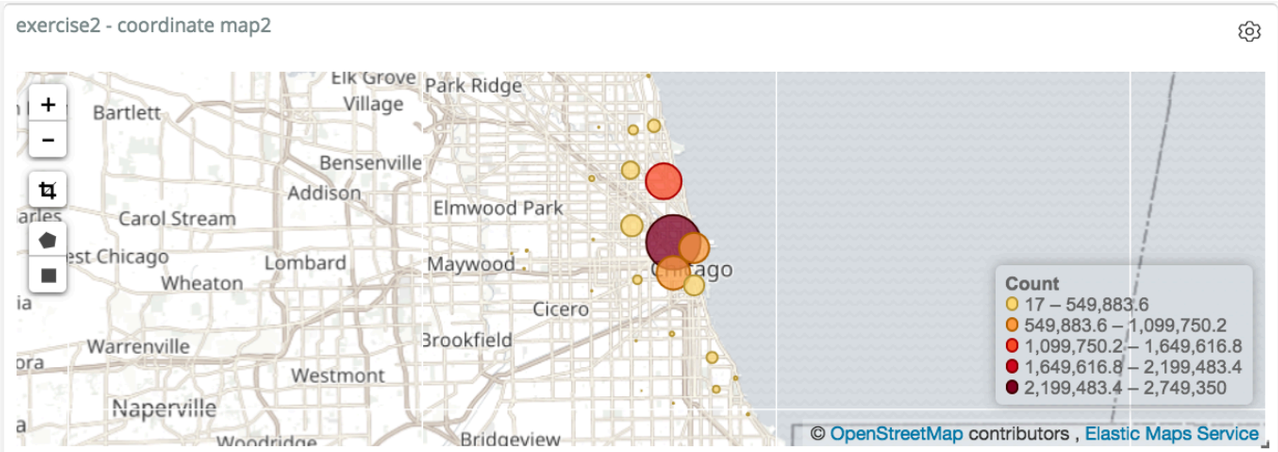
Visualize - Coordinate Map 👑

- Buckets : end_gps field를 기준으로 Geohash aggregation
- Metrics : Documents Count

start_gps

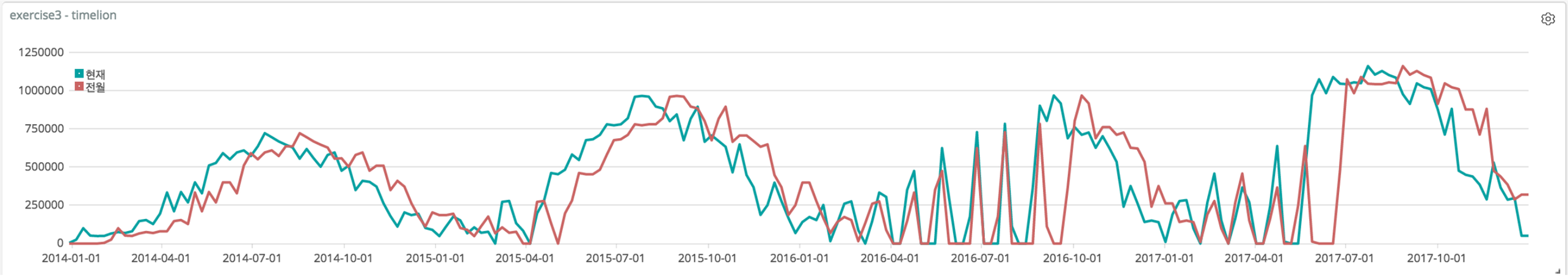
end_gps





Visualize - Timelion 👑

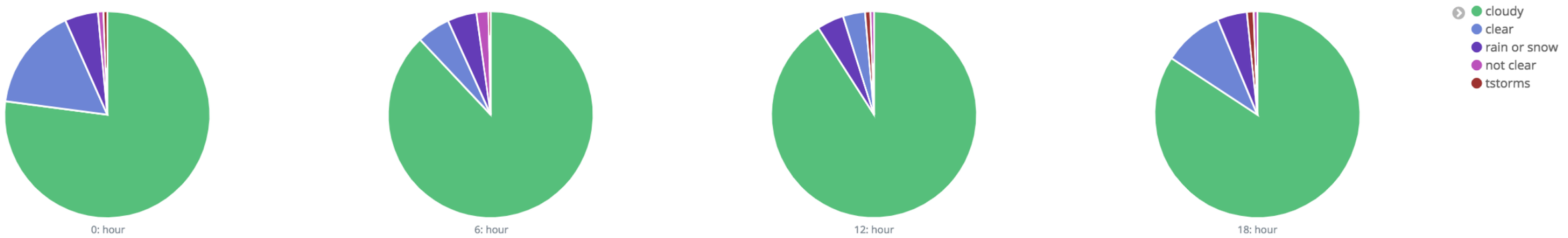
	Index	Timefield	Metric	Offset	Label
Series1	chicago*	starttime	tripduration의 sum	없음	현재
Series2				한 달 전	전월


exercise3 - timelion
현재
전월
1250000
1000000
750000
500000
250000
0
2014-01-01
2014-04-01
2014-07-01
2014-10-01
2015-01-01
2015-04-01
2015-07-01
2015-10-01
2016-01-01
2016-04-01
2016-07-01
2016-10-01
2017-01-01
2017-04-01
2017-07-01
2017-10-01

Visualize - Pie Chart 👑

- Buckets
  - Split Chart : hour field를 기준으로 interval이 6인 Histogram Aggregation 적용
  - Split Slices : events field를 기준으로 Documents Count가 많았던 상위 5개 Field 값 선정
- Metrics : Documents Count

exercise4 - pie chart



Visualize - Data Table 👑

- Buckets
  - Split Rows : from_station_name field를 기준으로 Documents Count가 많았던 상위 10개 값 선정
  - Split Rows : to_station_name field를 기준으로 Documents Count가 많았던 상위 10개 값 선정
- Metric
  - Metric : Count
  - Metric : tripduration field에 Average Aggregation 적용

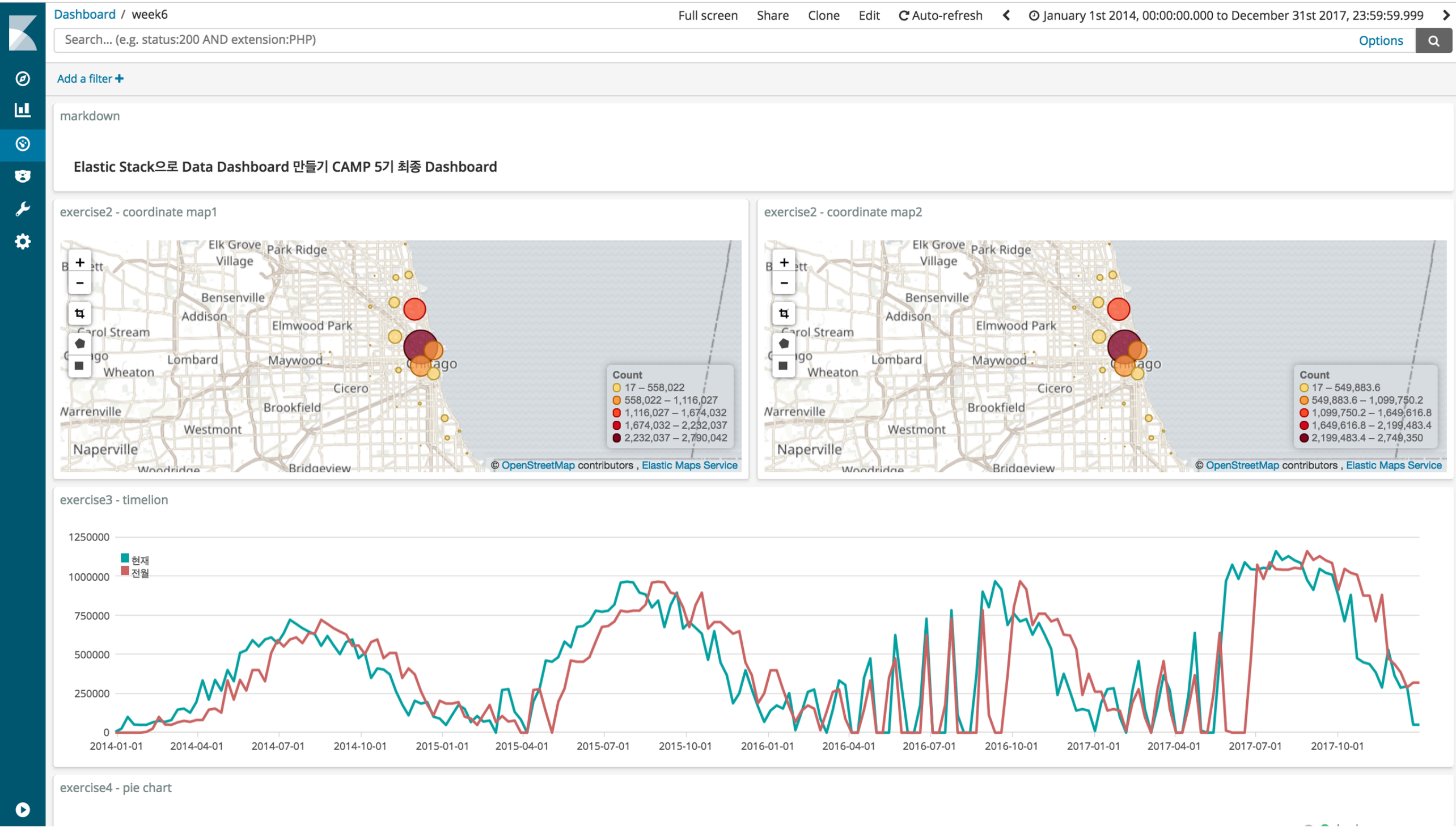
from_station	to_station	이용 건 수	평균 이용 시간
Clinton St & Washington Blvd	Michigan Ave & Washington St	5,666	7.902
Clinton St & Washington Blvd	LaSalle St & Jackson Blvd	3,843	6.745
Clinton St & Washington Blvd	Columbus Dr & Randolph St	3,708	11.204
Clinton St & Washington Blvd	Michigan Ave & Lake St	3,164	8.702
Clinton St & Washington Blvd	State St & Kinzie St	3,012	9.17
Clinton St & Washington Blvd	St. Clair St & Erie St	2,972	13.296
Clinton St & Washington Blvd	Morgan St & Lake St	2,025	5.842
Clinton St & Washington Blvd	Daley Center Plaza	1,878	6.655
Clinton St & Washington Blvd	Rush St & Hubbard St	1,861	9.109
Clinton St & Washington Blvd	Kingsbury St & Kinzie St	1,803	5.272

markdown

Elastic Stack으로 Data Dashboard 만들기 CAMP 5기 최종 Dashboard

Dashboard의 Title을 각자 짧게 입력하자

Visualizations을 적절히 배치해서 Dashboard를 만들자 👑



Query DSL를 이용해서 원하는 결과만 보자 👑

- must : to_station_name field가 Cli로 시작
- must not : from_station_name field = 313
- should (1개 이상)
    - events = clear
    - 30 ≤ dpcapacity_end ≤ 40

Logstash를 이용해서 elasticsearch 데이터를 csv로 추출하자 👑

input

- bool query
  - must
    - tags field가 VIP
    - gender field가 Female
    - 10 ≤ dpcapacity_end < 12
  - must_not : event fields가 cloudy
- sort : tripduration field 값이 큰 순으로 나열

output

- starttime
- stoptime
- tripduration

fields만 선택하여 csv 저장

3. 실습 데이터 백업 & 복원

방법은 여러가지가 있다

- python 등 client 이용 👑
- S3 등을 이용한 백업/복구 👑
- logstash 이용 👑
- reindex API 이용 👑

방법은 여러가지가 있다

- ~~python 등 client 이용~~ 👑
- ~~S3 등을 이용한 백업/복구~~ 👑
- **logstash** 이용 👑
- **reindex API** 이용 👑

방법은 여러가지가 있다 - logstash 👑(6.4.0 이상)

~~(그 아래 버전은 한글이 깨진다)~~

```
input {
  elasticsearch {
    hosts => ["elasticsearch.higee.co"]
    index => "shopping"
  }
}

output {
  elasticsearch {
    hosts => ["${host}"]
    index => "shopping-backup"
  }
}
```

default로 설정한 scroll (👑)과 size (👑) 옵션을 적절하게 이용하자

방법은 여러가지가 있다 - reindex API

Step 1

```
1 version: '3'
2
3 services:
4
5   elasticsearch:
6     image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.2.4
7     container_name: elasticsearch
8     environment:
9       http.host: '0.0.0.0'
10      network.host: '127.0.0.1'
11      ES_JAVA_OPTS: '-Xms2g -Xmx2g -XX:-AssumeMP'
12      bootstrap.memory_lock: 'true'
13      reindex.remote.whitelist: "elasticsearch.higee.co:80"
```

본인 elasticsearch.yml에 이 설정을 추가한 후 elasticsearch를 다시 시작해야 reindex API를 사용 가능

방법은 여러가지가 있다 - reindex API 👑

Step 2

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "http://elasticsearch.higee.co:80"
        },
    "index": "shopping",
    "query": {
      "match_all": {}
      }
    },
  "dest": {
    "index": "shopping-backup"
  }
}
```

reindex API의 remote 설정을 활성화하면 원격 elasticsearch data도 재색인 가능하다

방법은 여러가지가 있다 - reindex API

Step 3

Dev Tools

Console

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "http://elasticsearch.higee.co:80"
        },
    "index": "shopping",
    "query": {
      "match_all": {}
      }
    },
  "dest": {
    "index": "shopping-backup"
  }
}
```

```
{
  "took": 4966,
  "timed_out": false,
  "total": 20000,
  "updated": 0,
  "created": 20000,
  "deleted": 0,
  "batches": 20,
  "version_conflicts": 0,
  "noops": 0,
  "retries": {
    "bulk": 0,
    "search": 0
  },
  "throttled_millis": 0,
  "requests_per_second": -1,
  "throttled_until_millis": 0,
  "failures": []
}
```

기본적으로 1000개 단위의 **batch 작업이다**
size 옵션을 통해 변경 가능하다 (다음 page 참고)

reindex API의 remote 설정을 활성화하면 원격 elasticsearch data도 재색인 가능하다

방법은 여러가지가 있다 - reindex API 👑

Step 3`

Dev Tools

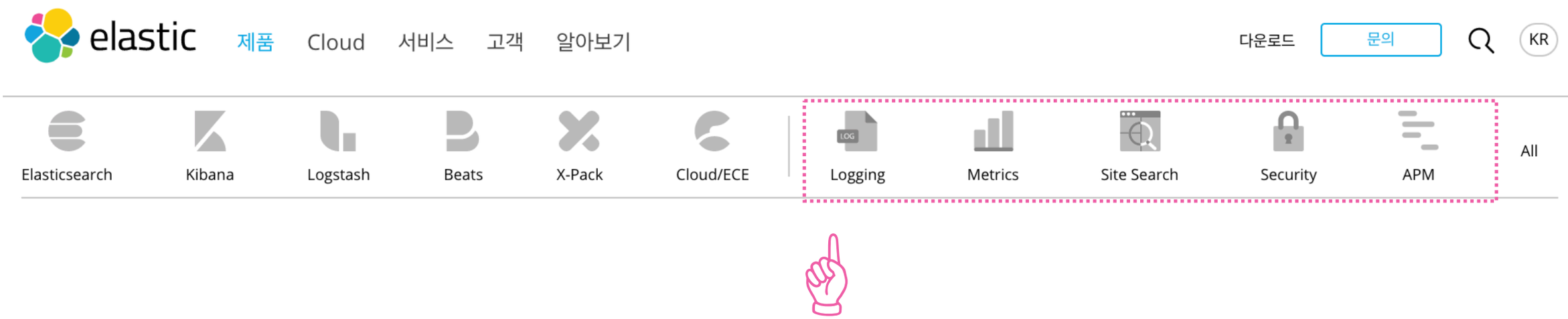
Console

```
POST _reindex
{
  "source": {
    "remote": {
      "host": "http://elasticsearch.higee.co:80"
       },
    "index": "shopping",
      "size": 2000,
    "query": {
      "match_all": {}
      }
    },
  "dest": {
    "index": "shopping-backup"
  }
}
```

```
{
  "took": 3967,
  "timed_out": false,
  "total": 20000,
  "updated": 0,
  "created": 20000,
  "deleted": 0,
  "batches": 10,
  "version_conflicts": 0,
  "noops": 0,
  "retries": {
    "bulk": 0,
    "search": 0
  },
  "throttled_millis": 0,
  "requests_per_second": -1,
  "throttled_until_millis": 0,
  "failures": []
}
```

4. 마무리

Elastic Stack의 활용처는 다양하다 👑



단,어떤 목적으로 사용하든 기본적인

Elastic Stack에 대한 이해도는 공통으로 필요하다.

그래서 수업 때 배운 내용을 충분히 이해했다는 전제 하에

다음과 같은 공부방향을 제시한다

Elastic Stack (조금만 더) 심층 Study

- Elasticsearch : Query DSL 👑, Aggregation 👑
- Logstash : 다양한 Input Plugin 👑/Output Plugin 👑/Filter Plugin 👑
- Kibana : 다양한 Visualization 👑
- Beats : 데이터 수집에 최적화된 가벼운 logstash 버전 : Filebeat, Metricbeat, … 👑
- (Curator) : Index와 Snapshot 단위 다양한 작업(delete, snapshot 등)을 정기적으로 수행 👑 (향후 Kibana 편입 예정)

검색엔진 (형태소 분석기)

- 어떤 한국어 분석기를 사용할까? 👑
- Elasticsearch에서 아리랑 한글 분석기 사용하기 👑

Elasticsearch Cluster 구축/운영

- Node 역할 👑
- Shard 및 Replica 개념 👑
- Life inside a Cluster (수평확장 및 장애대응) 👑

X-pack을 사용한다면…

- Anomaly Detection (Machine Learning) 👑
- Security (encryption, access control, …) 👑
- Alert System 👑 **<-> Open Source : ElastAlert** 👑
- APM (Application Performance Management) : Python, Node.js 👑 **(Basic License 👑)**

질문 및 Feedback은

gshock94@gmail.com로 주세요