Elastic Stack과 공공데이터를 활용한 서울시 지하철 대시보드 만들기

Elastic | Tech Evangelist |
김종민
2019-10-17

발표자 소개

Elasticsearch 한국 커뮤니티를 만들었고

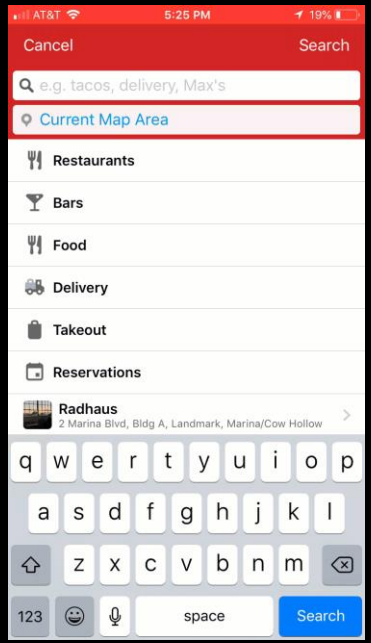
Elastic 의 첫 번째 한국 직원입니다

Elastic?

Elastic 은 검색엔진 회사입니다

Elastic?

Elastic 은 검색엔진 회사입니다



SOSCON2019

SAMSUNG OPEN SOURCE CONFERENCE 2019

Elastic?

검색엔진은 생각보다 많은 쓰임새가 있습니다.

Tech 기업의 딜레마 ...

일반적으로 컨퍼런스에서 솔루션 소개 등은 선호하지 않습니다.
서비스 기업의 운영 중 문제 해결 사례 등을 선호합니다.

처음 설치부터 클러스터 관리까지, 3000대의 운영 노하우로 만든 카카오 Elasticsearch 운영 방법론 소개와 자동화 솔루션 시연

카카오에서는 ElasticSearch 검색엔진부터 분석엔진까지 3000개 넘는 노드를 운영하고 있습니다. 이렇게 많은 노드들을 잘 운영하려면 무엇이 필요할까요?
레퍼런스 문서에 나오지 않는 검색 엔진 클러스터 최적의 성능을 찾아가는 방법, 초당 수억건의 문서를 저장하는 분석엔진 운영 방법, 장기간의 데이터를 가성비를 고려하여 보관하는 방법, ansible 이나 Elastic Stack, 스크립트 등을 활용하여 손쉽게 운영할 수 있는 방법까지!
200개가 넘는 클러스터의 레퍼런스를 바탕으로 로그분석, 메일검색, 카카오톡 # 탭 등의 클러스터 특성별 최적화 방안과 운영 노하우, 그 외에 카카오에서 ElasticSearch 를 대상으로 하고 있는 다양한 실험들을 최초로 공개합니다.

#ElasticSearch #설치부터_관리까지 #3000대_운영노하우 #자동화_솔루션 #다양한실험

밑바닥부터 시작하는 쇼핑 데이터 엔지니어링 고군 분투기

Scale-Up 한계에 부딪힌 DB를 Elasticsearch로 전환하면서 고민했던 기술 노하우를 공유합니다. DB 데이터 변경 이벤트를 잡아내는 기본 원리를 소개하고, 변경된 이벤트로 검색 서비스를 만들기 위해 구성했던 데이터 파이프라인(실시간과 배치처리)에 대해서 이야기합니다. 데이터 정합성을 위한 아키텍쳐, 개발 테스트 환경 및 운영 전략과 간단한 팁에 대해서도 이야기합니다.

#ElasticSearch #실시간처리 #Spark #Kafka #데이터파이프라인

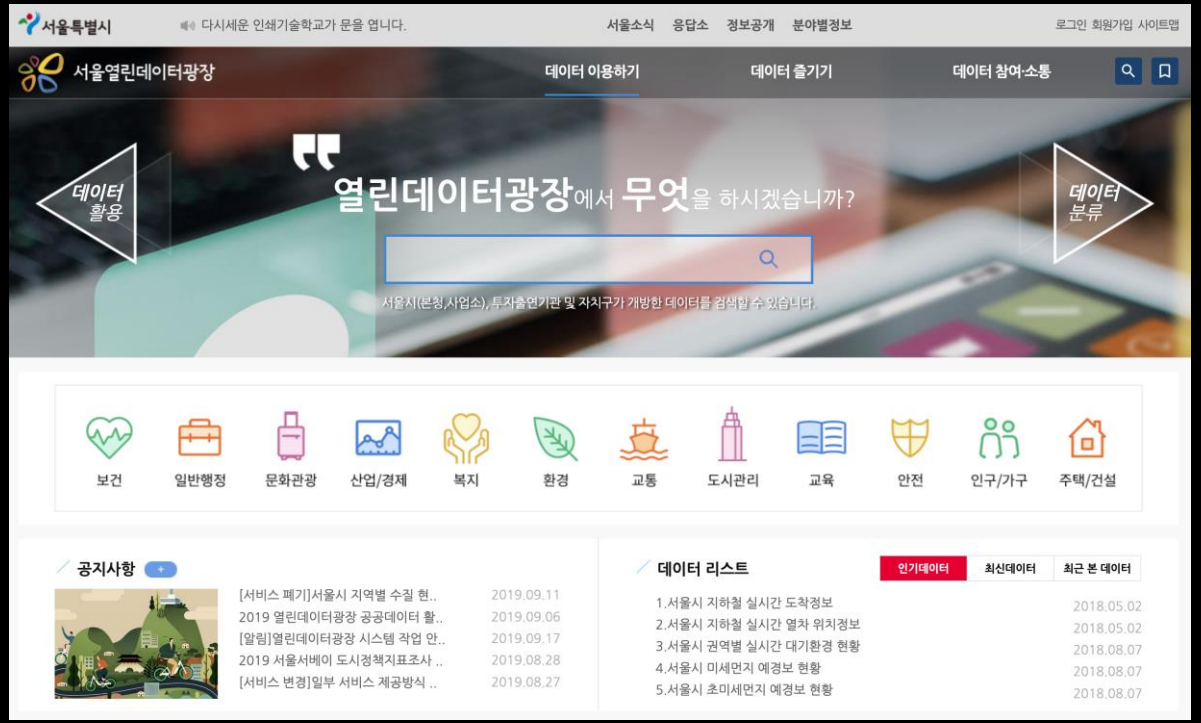
Tech 기업의 딜레마 ...

어디선가 발표를 하려면 스토리, 멋진 데모가 있어야 합니다.

Elastic 은 도구를 만드는 곳이지 서비스 기업 아니기 때문에 데이터를 직접 구해야 합니다.

그래서 스토리를 만들려면 항상 골치가 아픕니다.

종종 이런 곳을 기웃거리게 됩니다.



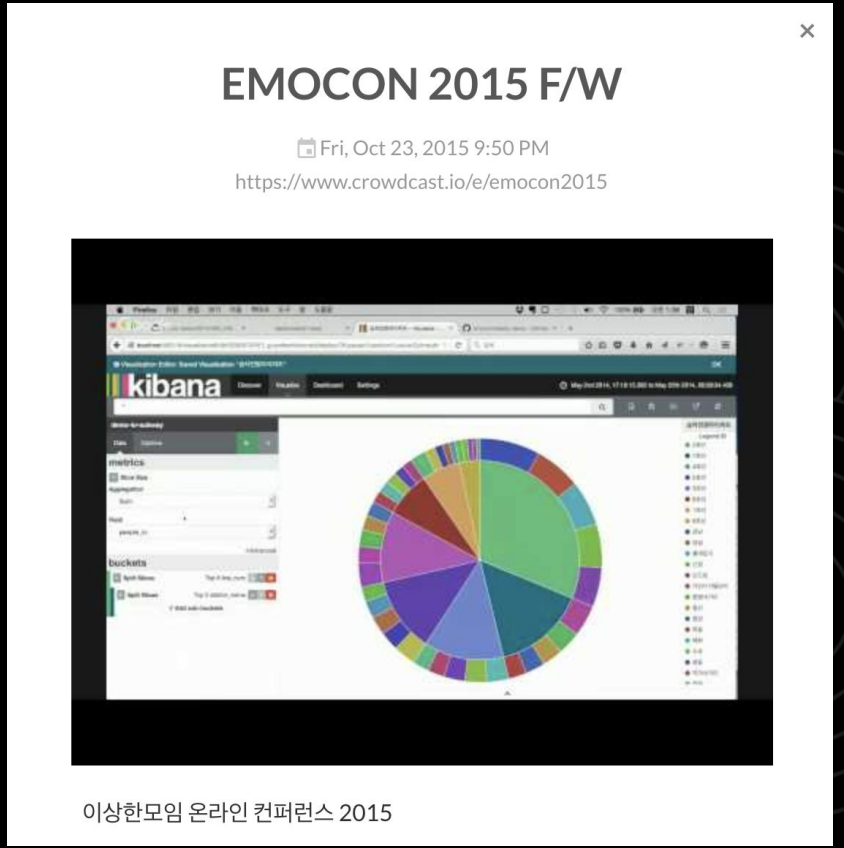
2015년 10월, 아직 Elastic 이 많이 알려지지 않았을 무렵

Elastic 을 알리기 위해 있는 행사 없는 행사 다 찾아다니던 중

이상한 모임 이라는 온라인 커뮤니티에서 온라인으로 하는 이상한 컨퍼런스에서 발표를 하게 됩니다

Elasticsearch 와 Kibana 로 서울시 지하철 대시보드 만들기를 소개

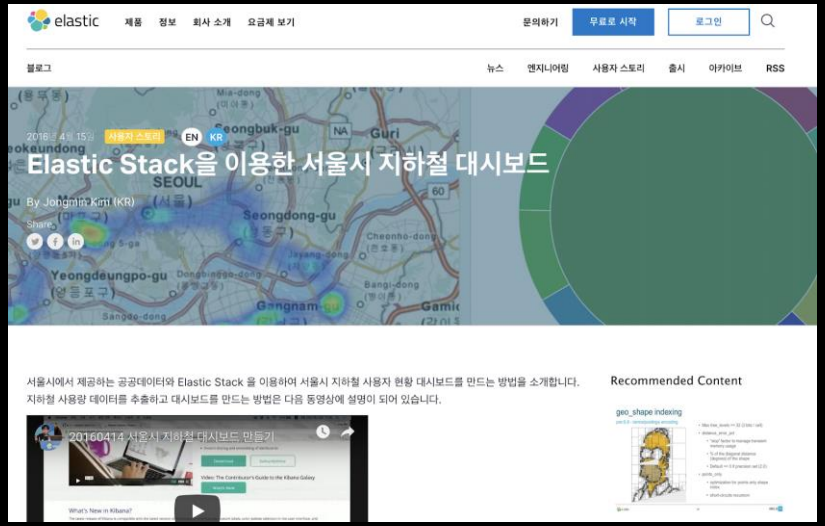
상당한 호평을 받았습니다

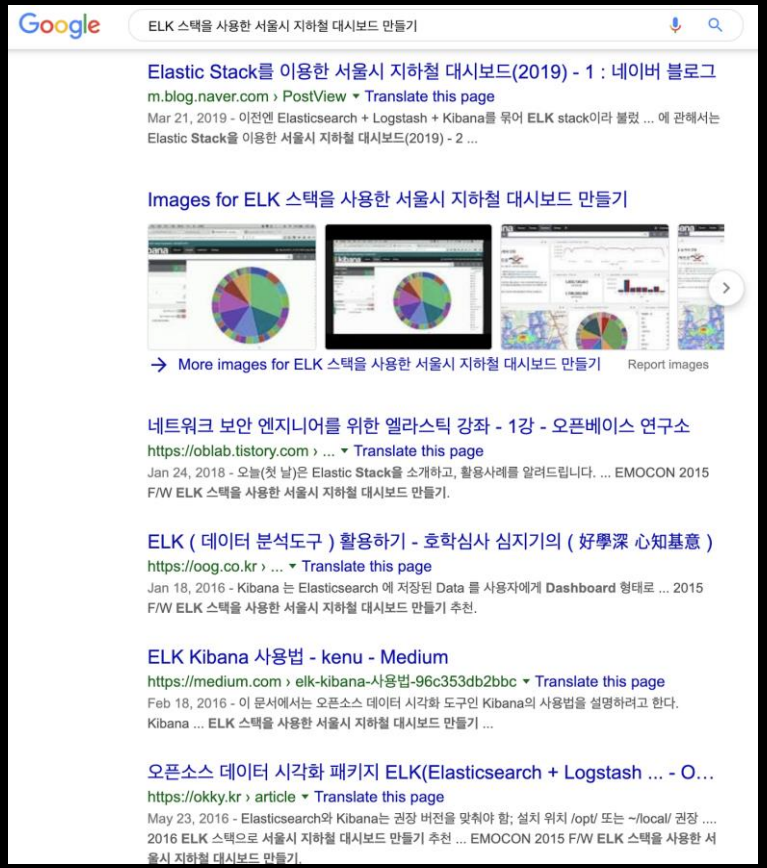


발표 이후로도

Elastic 에 대한 관심은 점점 높아지고 있었는데,
당시 소개 된 자료가 많지 않아 이 데모가 널리
유용하게 씌였습니다.

공식 홈페이지에도 블로그로 업로드 되었습니다.





한번은 이런 적도...

어떤 초보 개발자분이 인턴 기간 중에 Elastic – Kibana 서울 지하철 대시보드 만들기 데모를 보고
회사에서 이 내용으로 발표를 해서 나중에 정직원으로 취업이 되었다는 감사 메일도 받았습니다.

On 2016년 10월 11일 at 오후 5:18:23, [redacted] wrote:

안녕하세요?

저는 [redacted]고하며, [redacted]에서 인턴 3개월차에 접어든 초보 개발자입니다.

빅데이터 시각화를 맡아 kibana를 공부하던 중에 김종민님의 서울시 지하철 대시보드 데모를 보게 됐습니다.

ELK 설정부터 차근차근 대시보드 시각화까지 설명해주셔서 이 데모를 따라하는 것만으로도 많은 공부가 될 것이라 생각했습니다.

김종민 개발자님 안녕하세요?

[redacted] 9개월간의 노력의 결실을 맺는 순간이라 매우 기쁩니다!^^

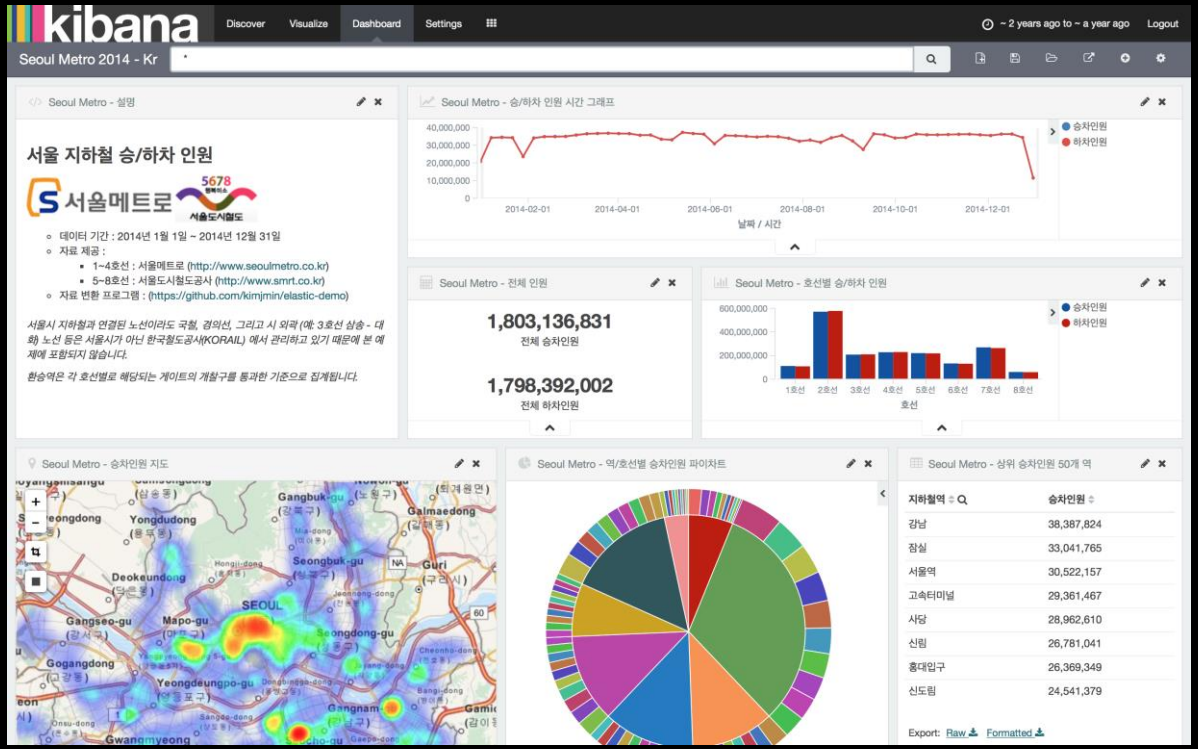
인턴기간동안 김종민 개발자님과 서신을 주고받으며 ELK를 열심히 공부했더니, 사원 첫 프로젝트에서도 ELK 관련 담당을 하게 되네요. 아차, '시작하세요! 엘라스틱서치'를 보면서 제가 김종민 개발자님께 물어봤던 많은 질문의 답을 찾을 수 있었습니다. 책을 자세하게 적어주셔서 감사합니다.

그 이후로도 오랬동안

이 데모는 여러 행사에서 단골 예제로 쓰이게 됩니다.

사실 이 이상 잘 되어 있는 데이터셋을 찾기도 쉽지 않았습니다.

우리 생활에 밀접하게 관련 되어 있는 데이터라 설명 하기도 쉬웠습니다.



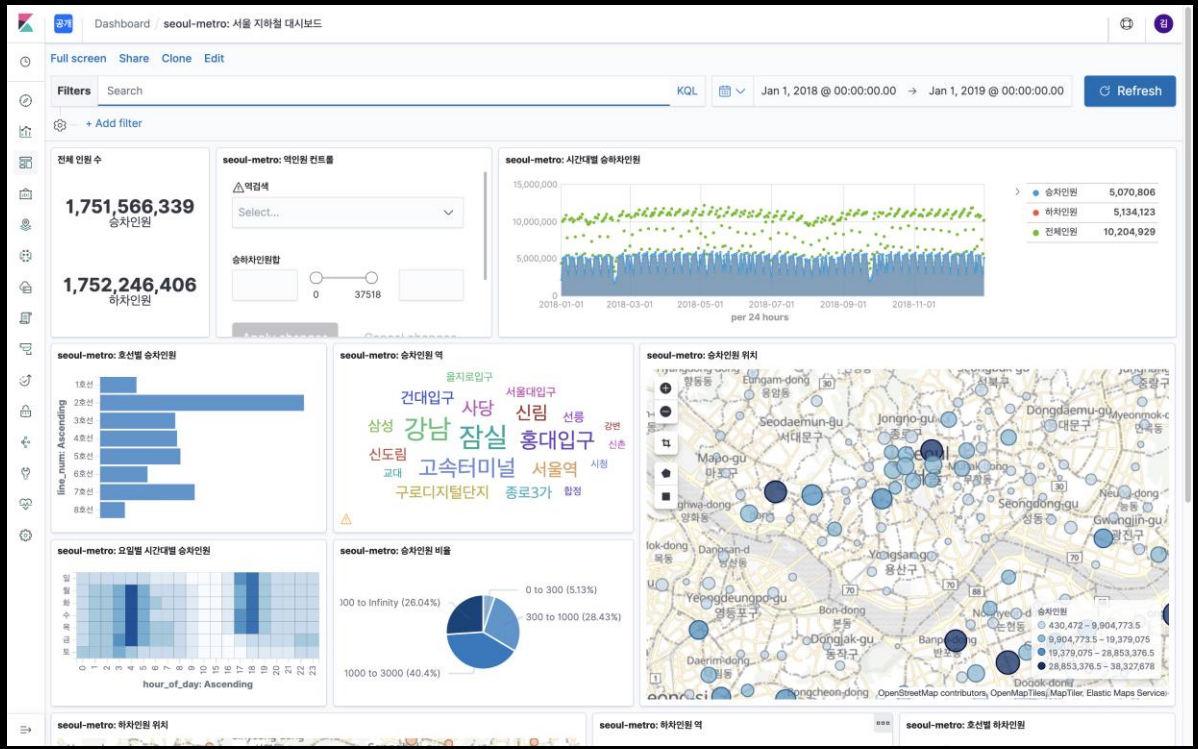
시간은 흘러

Elastic 은 계속 새 버전이 나왔고, 새로운 데이터도 계속 나왔습니다.

처음 데모 만들 때가 2.4
현재는 7.4 버전 까지
나왔습니다.

지하철 데이터셋도
계속 업데이트 되었습니다.

- ~2018 년 데이터 까지
- 역명 다국어 정보
- 지하철 역 위치정보

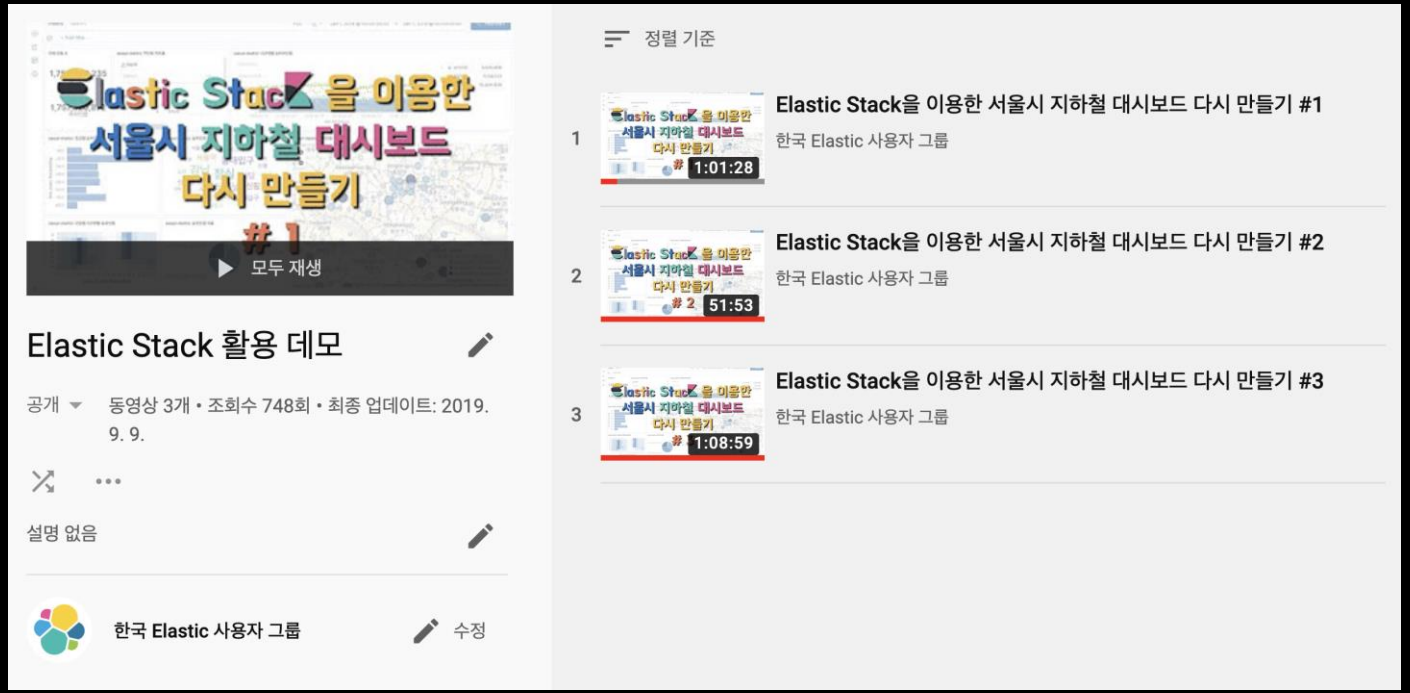


전체 과정

https://github.com/eskrug/elastic-demos

위의 깃헙 리파지토리에 자세히 설명하고 있습니다

유튜브 비디오도 있습니다



활용 데이터셋 - https://data.seoul.go.kr

1. 서울시 역코드로 지하철역 위치 조회

전철역코드, 전철역명, 호선, 위도, 경도 값 등을 포함하고 있습니다.

필터선택

EXCEL CSV JSON

전철역코드	전철역명	호선	외부코드	사이버스테이션	X좌표	Y좌표	X좌표(WGS)	Y좌표(WGS)
0153	종로3가	1	130	0153	498060	1130332	37.571607	126.991806
0155	동대문	1	128	0155	502155	1130180	37.571420	127.009745
0150	서울	1	133	0150	493365	1125595	37.554648	126.972559
1007	신도림	1	140	0234	475755	1112937	37.508725	126.891295
0151	시청	1	132	0151	494587	1128532	37.564718	126.977108

json 형태의 값을 내려받아 사용합니다.


```
'X좌표(WGS)","STATION_NM":"전철역명","STATION_CD":"전철역코드","LINE_NUM":"호선","FR_CODE":"외부코드","CYBER_ST_CODE":"사이버스테이션","XPOINT":"X좌표

vgs":"37.492522","ypoint":"1108579","ypoint_wgs":"127.118234","xpoint":"525992","station_cd":"2818","station_nm":"가락시장","fr_code":"817"},
vgs":"37.492522","ypoint":"1108579","ypoint_wgs":"127.118234","xpoint":"525992","station_cd":"0340","station_nm":"가락시장","fr_code":"350"},
vgs":"37.571607","ypoint":"1130332","ypoint_wgs":"126.991806","xpoint":"498060","station_cd":"2535","station_nm":"종로3가","fr_code":"534"},
vgs":"37.571607","ypoint":"1130332","ypoint_wgs":"126.991806","xpoint":"498060","station_cd":"0319","station_nm":"종로3가","fr_code":"329"},
```

2. 서울교통공사 지하철 역명 다국어 표기 정보

지하철 역명의 한글, 한자, 영문, 중국어, 일본어를 포함하고 있습니다.

한글	한자	영문	중국어	일본어
강남구청	江南區廳	Gangnam-gu …	江南区厅	カンナムグチョン
강동	江東	Gangdong	江东	カンドン
강동구청	江東區廳	Gangdong-gu …	江东区厅	カンドングチョン
개롱	開籠	Gaerong	开笼	ケロン
개화산	開花山	Gaehwasan	开花山	ケファサン

json 형태의 값을 내려받아 사용합니다.


```
{
"DESCRIPTION" : {"STN_NM_ENG":"영문","STN_NM_CHN":"중국어","STN_NM":"한글","STN_NM_CHC":"한자","STN_NM_JPN":"일본어"},
"DATA" : [
{"stn_nm_chn":"可乐市场","stn_nm":"가락시장","stn_nm_eng":"Garak Market","stn_nm_chc":"可樂市場","stn_nm_jpn":"カラク·シジャン"},
{"stn_nm_chn":"加山数码园区","stn_nm":"가산디지털단지","stn_nm_eng":"Gasan Digital Complex","stn_nm_chc":"加山디지털團地","stn_nm_jpn":"カ
{"stn_nm_chn":"江南区厅","stn_nm":"강남구청","stn_nm_eng":"Gangnam-gu Office","stn_nm_chc":"江南區廳","stn_nm_jpn":"カンナムグチョン"},
{"stn_nm_chn":"江东","stn_nm":"강동","stn_nm_eng":"Gangdong","stn_nm_chc":"江東","stn_nm_jpn":"カンドン"},
{"stn_nm_chn":"江东区厅","stn_nm":"강동구청","stn_nm_eng":"Gangdong-gu Office","stn_nm_chc":"江東區廳","stn_nm_jpn":"カンドングチョン"},
```

3. 서울교통공사 연도별 일별 시간대별 역별 승하차 인원

년도별 일별 시간대별 역별 승하차 인원을 excel, csv 형태로 제공합니다.

서울교통공사 연도별 일별 시간대별 역별 승하차 인원관련 파일을 제공합니다. 다운로드 하세요.

NO	파일명	파일크기(KB)	마지막수정일	최초공개일	다운로드
1	서울교통공사 2008년 일별 역별 시간대별 승하차인원 (1~8호선).csv	28870.97	2017.08.28	2017.08.28	DOWN
2	서울교통공사 2009년 일별 역별 시간대별 승하차인원 (1~8호선).csv	27948.83	2017.08.28	2017.08.28	DOWN

	A	B	C	D	E	F	G	H	I	J	K	L
1	날짜	호선	역번호	역명	구분	05~06	06~07	07~08	08~09	09~10	10~11	11~12
2	2017-01-01	1	150	서울역	승차	470	286	397	786	1421	1459	2623
3	2017-01-01	1	150	서울역	하차	278	880	859	964	1407	1622	1681
4	2017-01-01	1	151	시청	승차	204	105	112	162	288	396	431
5	2017-01-01	1	151	시청	하차	73	203	314	483	669	611	668
6	2017-01-01	1	152	종각	승차	791	390	245	270	323	482	767
7	2017-01-01	1	152	종각	하차	61	184	254	676	1031	992	1084

데이터셋 조합

1. 위치 데이터와 다국어 데이터 조합 -> 메타 정보 데이터 생성

- "역명" 을 기준으로 병합합니다.
- 역명이 달라서 병합이 되지 않는 건 들은 동일한 역명으로 맞추어 주어야 합니다


```
"station_cd":"2533","station_nm":"서대문","fr_code":"532"},
"station_cd":"2534","station_nm":"광화문","fr_code":"533"},
"station_cd":"2635","station_nm":"청구","fr_code":"634"},
```

```
大学)","stn_nm":"광나루\n(장신대)","stn_nm_eng":"Gwa
,"stn_nm":"광명사거리","stn_nm_eng":"Gwangmyeongsage
文化会馆)","stn_nm":"광화문\n(세종문화회관)","stn_nm
)","stn_nm":"광흥창\n(서강)","stn_nm_eng":"Gwangheur
```


```
// 광나루\n(장신대) 같은 줄바꿈 이름 앞 이름만 따옴.
if(sNames[i].stn_nm.indexOf("\n") > 0){
  stn_nm_short = sNames[i].stn_nm.split("\n")[0];
} else {
  stn_nm_short = sNames[i].stn_nm;
}

// 총신대입구(이수) 같은 "(" 앞 이름만 따옴
if(sNames[i].stn_nm.indexOf("(") > 0){
  stn_nm_short = sNames[i].stn_nm.split("(")[0];
} else {
  stn_nm_short = sNames[i].stn_nm;
}
```

데이터셋 조합

1. 위치 데이터와 다국어 데이터 조합 -> 메타 정보 데이터 생성

- 위/경도 정보가 있는 위치정보 파일을 기준으로 루프를 돌며 병합합니다.
- 다국어 정보가 없는 경우 한글 정보만 남겨둡니다

```
'서대문':
 { stn_nm: '서대문',
   stn_nm_kor: '서대문',
   stn_nm_chc: '西大門',
   stn_nm_eng: 'Seodaemun',
   stn_nm_chn: '西大门',
   stn_nm_jpn: 'ソデムン',
   geo_x: 37.565773,
   geo_y: 126.966641 },
'광화문': { stn_nm: '광화문', geo_x: 37.571026, geo_y: 126.976669 },
```

데이터셋 조합

2. 로그 데이터 시간대별로 분리 + 메타 정보와 병합

- 05~06, 06~07, 시간대별로 되어 있는 열을 행으로 분리

```
// csv 파일 형태는 아래와 같이 되어야 함.
// 0           1     2       3      4    5       6       7       ... 23
// 날짜,       호선,  역번호, 역명,  구분, 05~06,  06~07,  07~08,  ... 23~24,
// 2018-01-01, 1호선, 150,    서울역, 승차, 373,    318,    365,    ... 781,

// 2줄씩 루프 돌면서 0~3 열 까지의 데이터가 동일한지 확인
for(var cd=1; cd< csv_data.length ; cd+=2){
  var dataIn = csv_data[cd];
  var dataOut = csv_data[cd+1];
  if(dataIn[0]===dataOut[0] && dataIn[1]===dataOut[1]
    && dataIn[2]===dataOut[2] && dataIn[3]===dataOut[3]){
    // 역명
    var station_name = dataIn[3];
    // 날짜
    var ldateTemp = dataIn[0].split('-');
    // 시간 값으로 루프
    for(var h=0; h < 20; h++){
      var ldate = new Date(ldateTemp[0],Number(ldateTemp[1])-1,ldateTemp[2],h);
      // 승차인원
      var people_in = dataIn[5+h];
      people_in = Number(people_in);
```

```
s_logs = {
  "@timestamp" : ldate,
  "code": dataIn[2],
  "line_num" : dataIn[1],
  "line_num_en" : line_num_lang[dataIn[1]],
  "station": {
    "name" : stn_nm_full,
    "kr" : s_meta[t_st_nm].stn_nm_kor,
    "en" : s_meta[t_st_nm].stn_nm_eng,
    "chc" : s_meta[t_st_nm].stn_nm_chc,
    "ch" : s_meta[t_st_nm].stn_nm_chn,
    "jp" : s_meta[t_st_nm].stn_nm_jpn
  },
  "location" : {
    "lat" : s_meta[t_st_nm].geo_x,
    "lon" : s_meta[t_st_nm].geo_y
  },
  "people":{
    "in" : people_in,
    "out" : people_out,
    "total" : people_in+people_out
  }
}
```

데이터셋 조합

앞의 세 종류의 데이터셋을 조합해서 다음과 같은 형태의 데이터를


```
{"@timestamp":"2017-01-01T09:00:00.000Z","code":"2532","line_num":"5","station":{"name":"충정로"}
,"location":{"lat":37.559973,"lon":126.963672},"people":{"in":14,"out":56,"total":70}}
{"@timestamp":"2017-01-01T10:00:00.000Z","code":"2532","line_num":"5","station":{"name":"충정로"}
,"location":{"lat":37.559973,"lon":126.963672},"people":{"in":0,"out":13,"total":13}}
{"@timestamp":"2016-12-31T15:00:00.000Z","code":"2533","line_num":"5","station":{"name":"서대문",
"kr":"서대문","en":"Seodaemun","chc":"西大門","ch":"西大门","jp":"ソデムン"},"location":{"lat":37
.565773,"lon":126.966641},"people":{"in":50,"out":12,"total":62}}
{"@timestamp":"2016-12-31T16:00:00.000Z","code":"2533","line_num":"5","station":{"name":"서대문",
"kr":"서대문","en":"Seodaemun","chc":"西大門","ch":"西大门","jp":"ソデムン"},"location":{"lat":37
.565773,"lon":126.966641},"people":{"in":69,"out":112,"total":181}}
{"@timestamp":"2016-12-31T17:00:00.000Z","code":"2533","line_num":"5","station":{"name":"서대문",
"kr":"서대문","en":"Seodaemun","chc":"西大門","ch":"西大门","jp":"ソデムン"},"location":{"lat":37
.565773,"lon":126.966641},"people":{"in":109,"out":127,"total":236}}
```

데이터 색인

Logstash 를 이용해서 데이터를 Elasticsearch 로 색인합니다

Expanded document

Table JSON

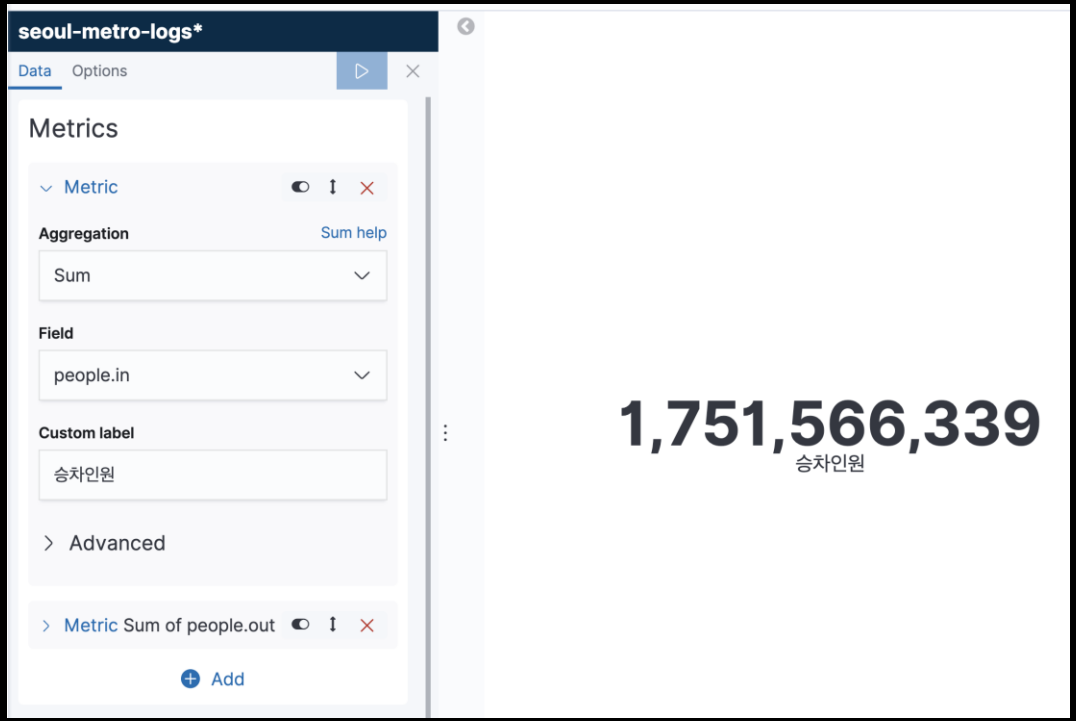

```
{
  "_index": "seoul-metro-logs-2018",
  "_type": "_doc",
  "_id": "cz7E32wBYMzov-8RWwBq",
  "_version": 1,
  "_score": null,
  "_source": {
    "line_num_en": "Line 1",
    "line_num": "1호선",
    "location": {
      "lon": 126.977108,
      "lat": 37.564718
    },
    "station": {
      "name": "시청"
    },
    "code": "151",
    "@timestamp": "2018-12-31T10:00:00.000Z",
    "people": {
      "in": 5730,
      "total": 6882,
      "out": 1152
    }
  },
  "fields": {
    "@timestamp": [
      "2018-12-31T10:00:00.000Z"
    ]
```


Expanded document

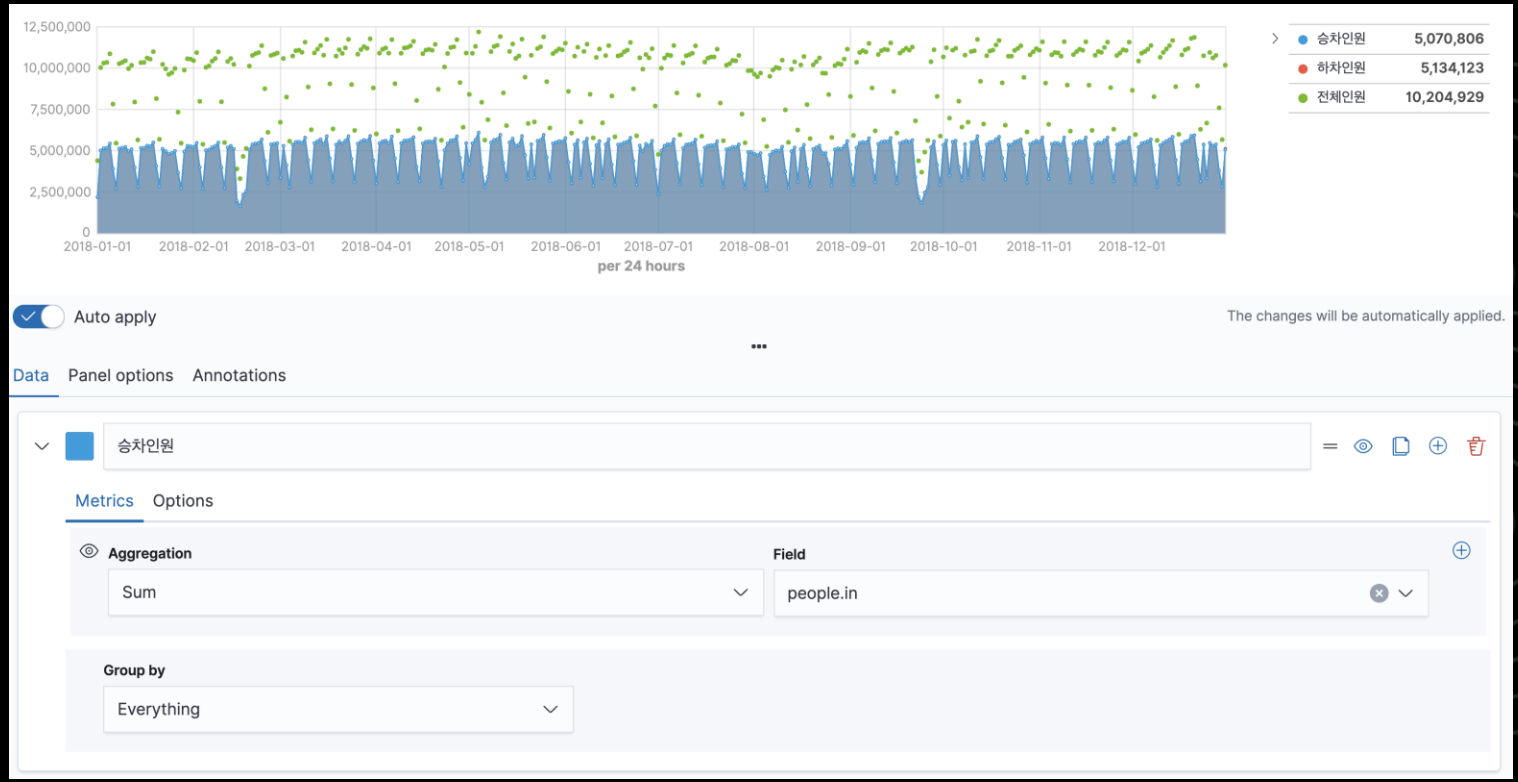
Table JSON

@	@timestamp	Dec 31, 2018 @ 19:00:00.000
t	_id	cz7E32wBYMzov-8RWwBq
t	_index	seoul-metro-logs-2018
#	_score	-
t	_type	_doc
t	code	151
t	line_num	1호선
t	line_num_en	Line 1
	location	{ "lon": 126.977108, "lat": 37.564718 }
#	people.in	5,730
#	people.out	1,152
#	people.total	6,882
t	station.name	시청

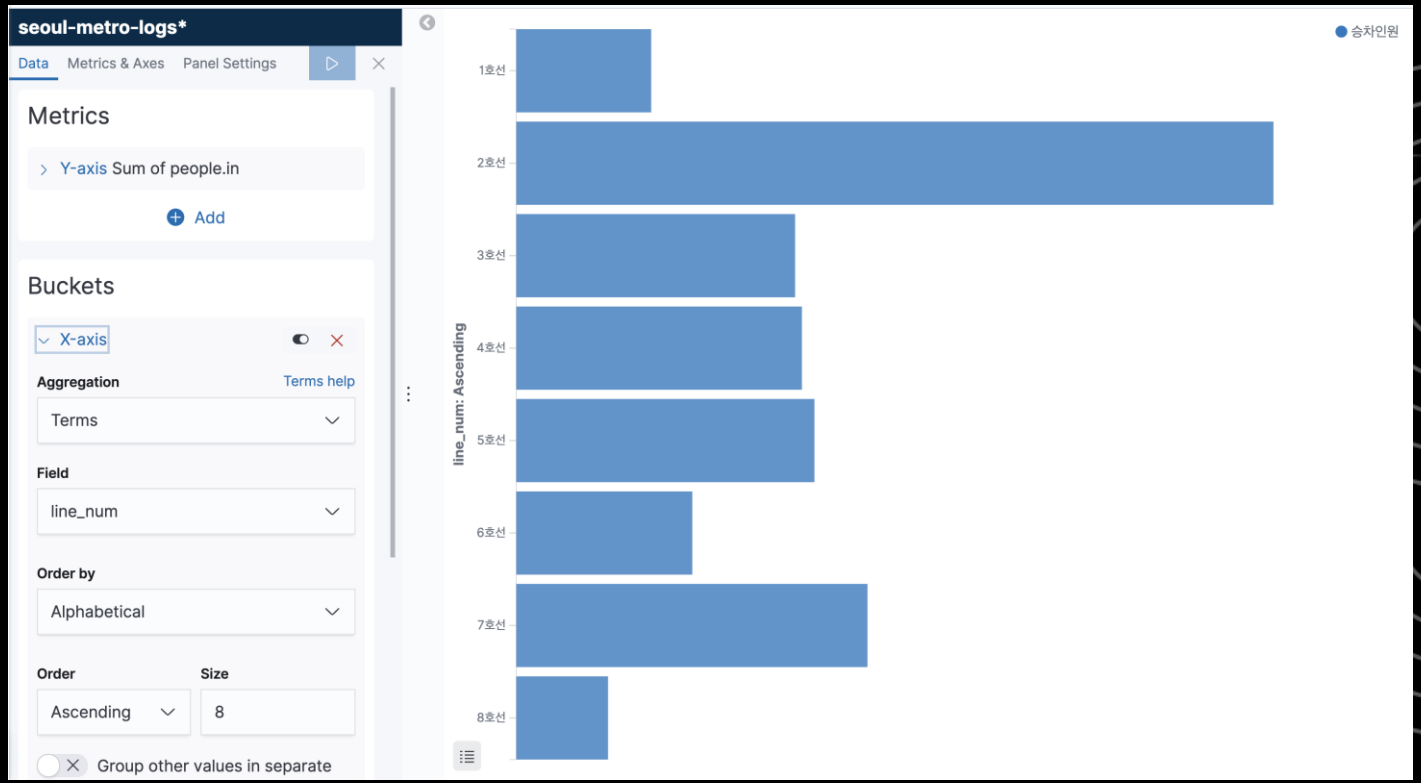
Metric : 전체 승하차 인원 수



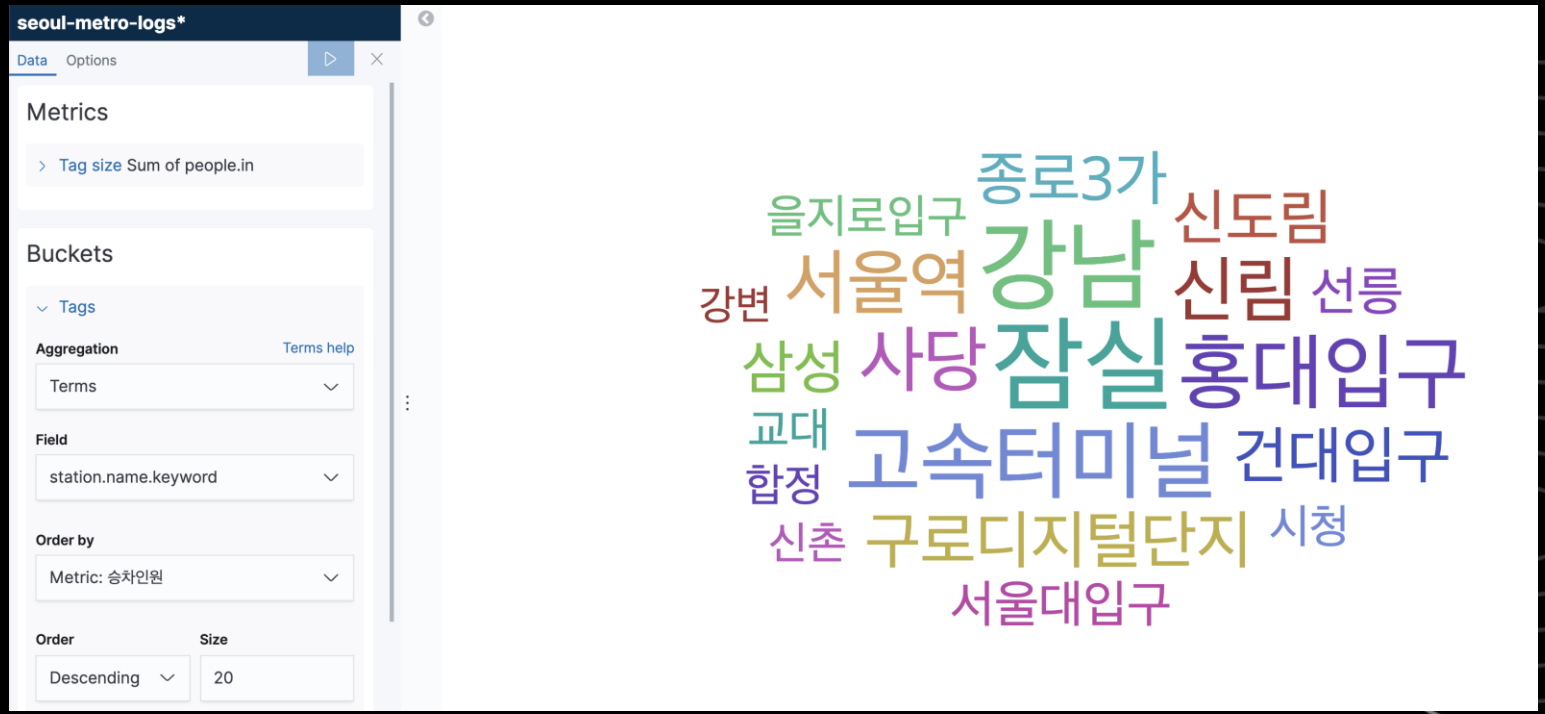
Time Series Visualization : 시간대별 승하차인원



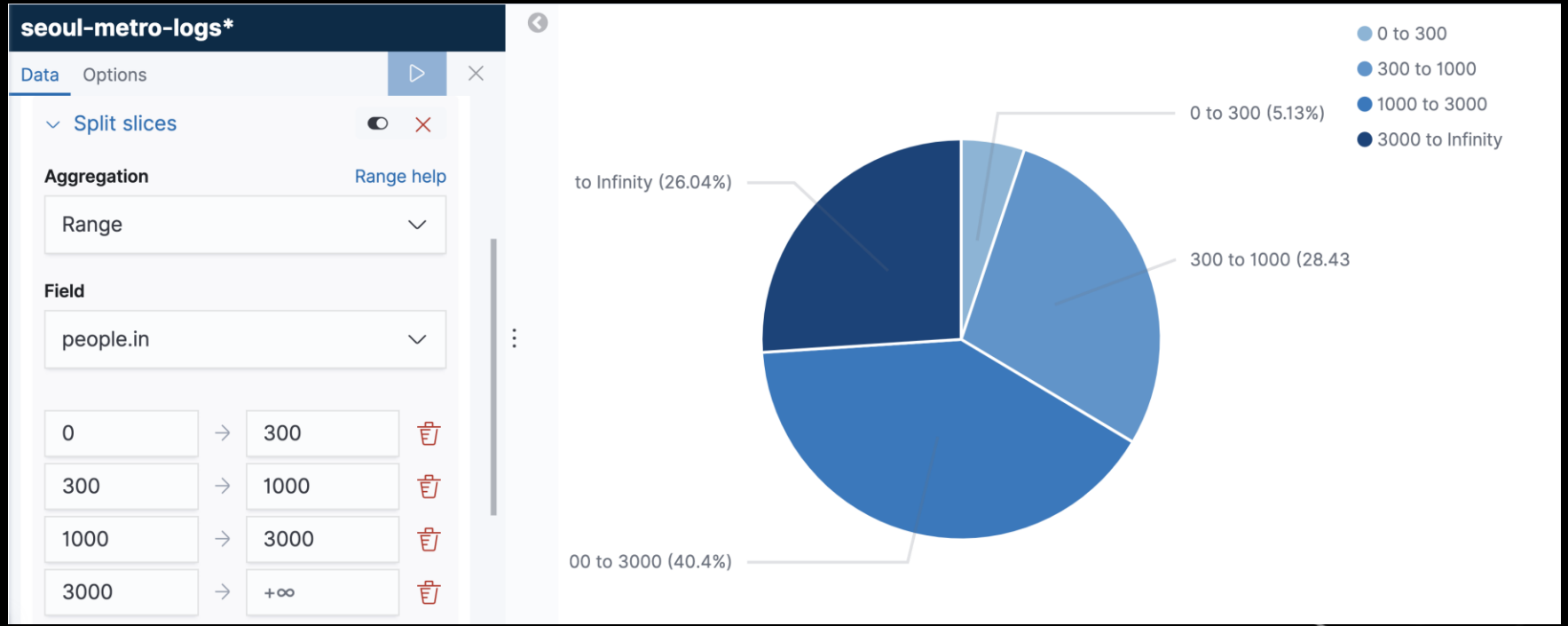
Bar Chart : 호선별 승하차인원



Tag Cloud : 역별 승하차 인원

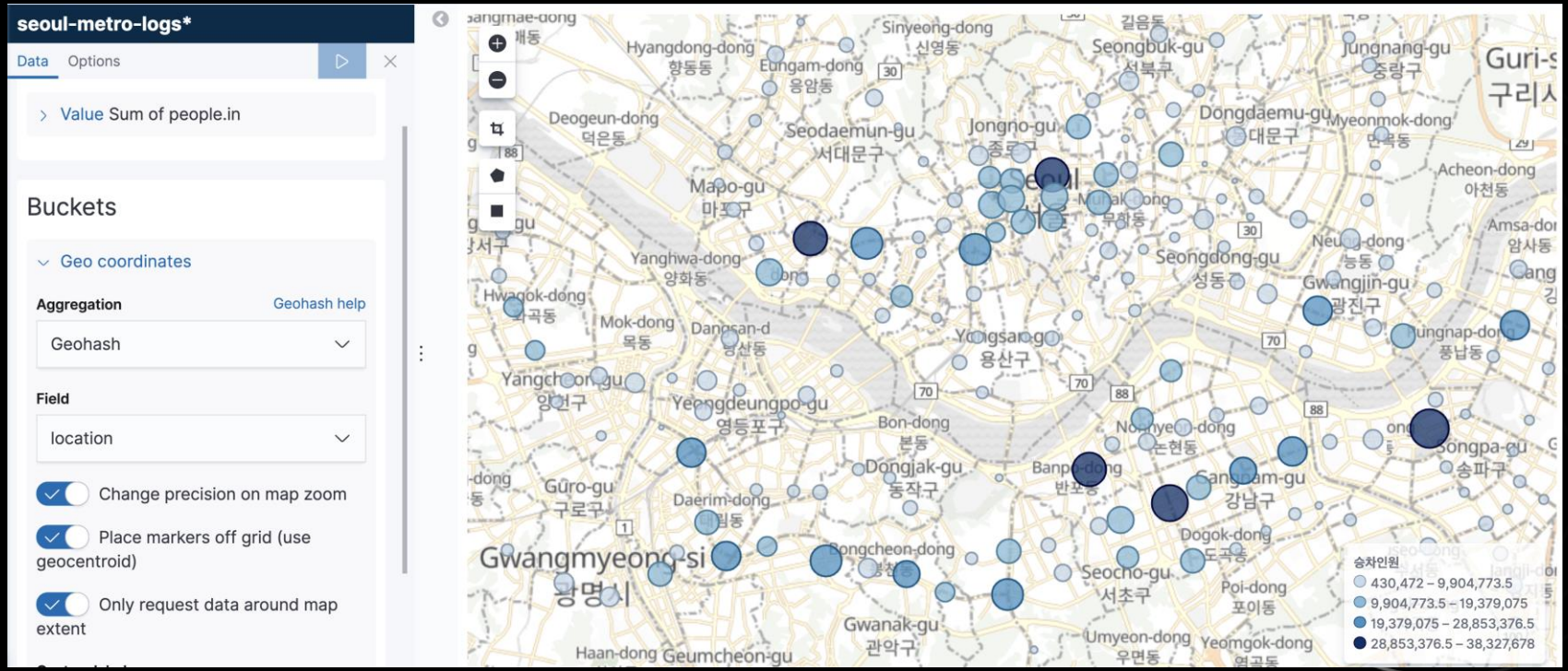


Pie Chart : 승하차인원 수 비율

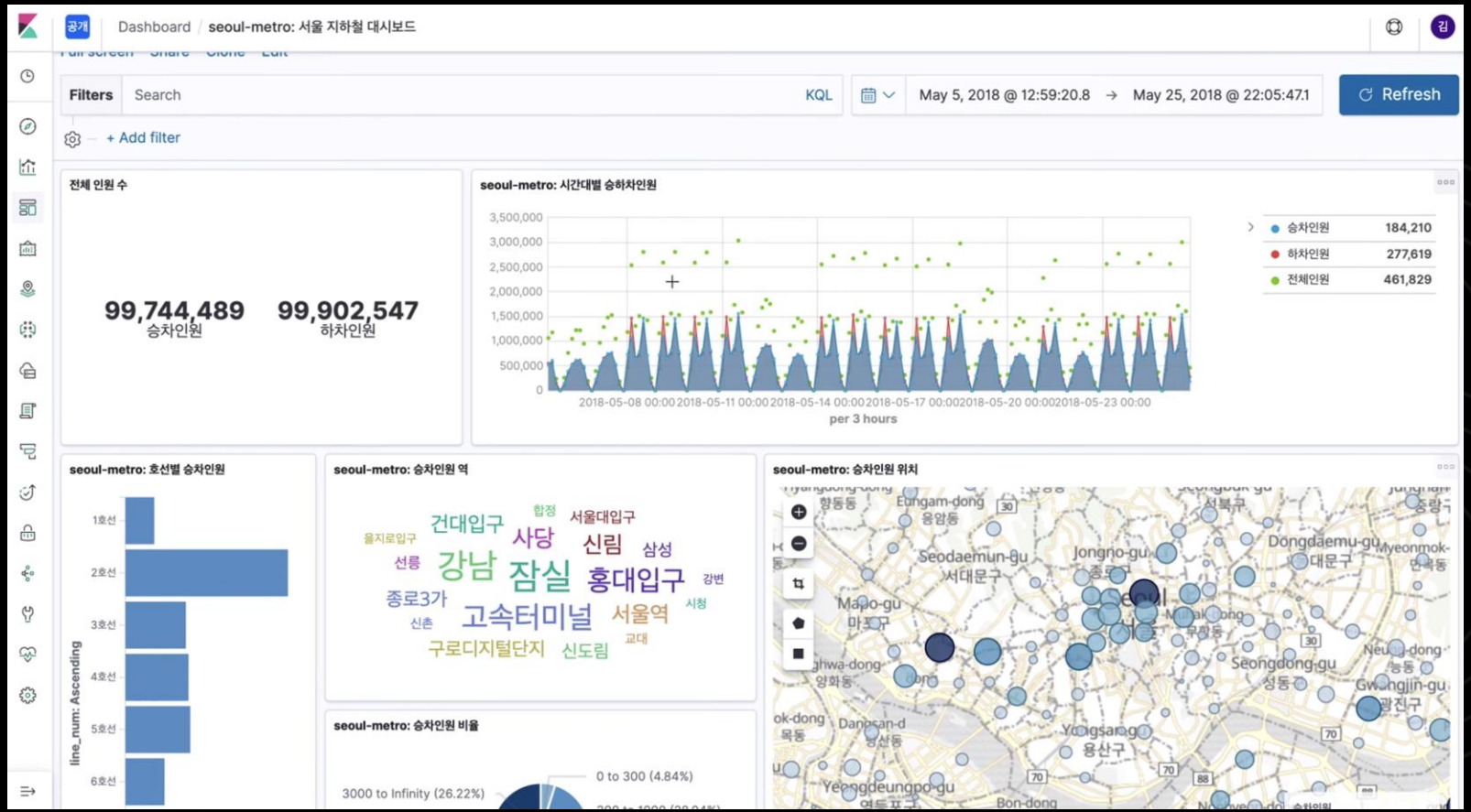


Kibana 대시보드

Map : 승하차인원 위치



1차 Kibana 대시보드

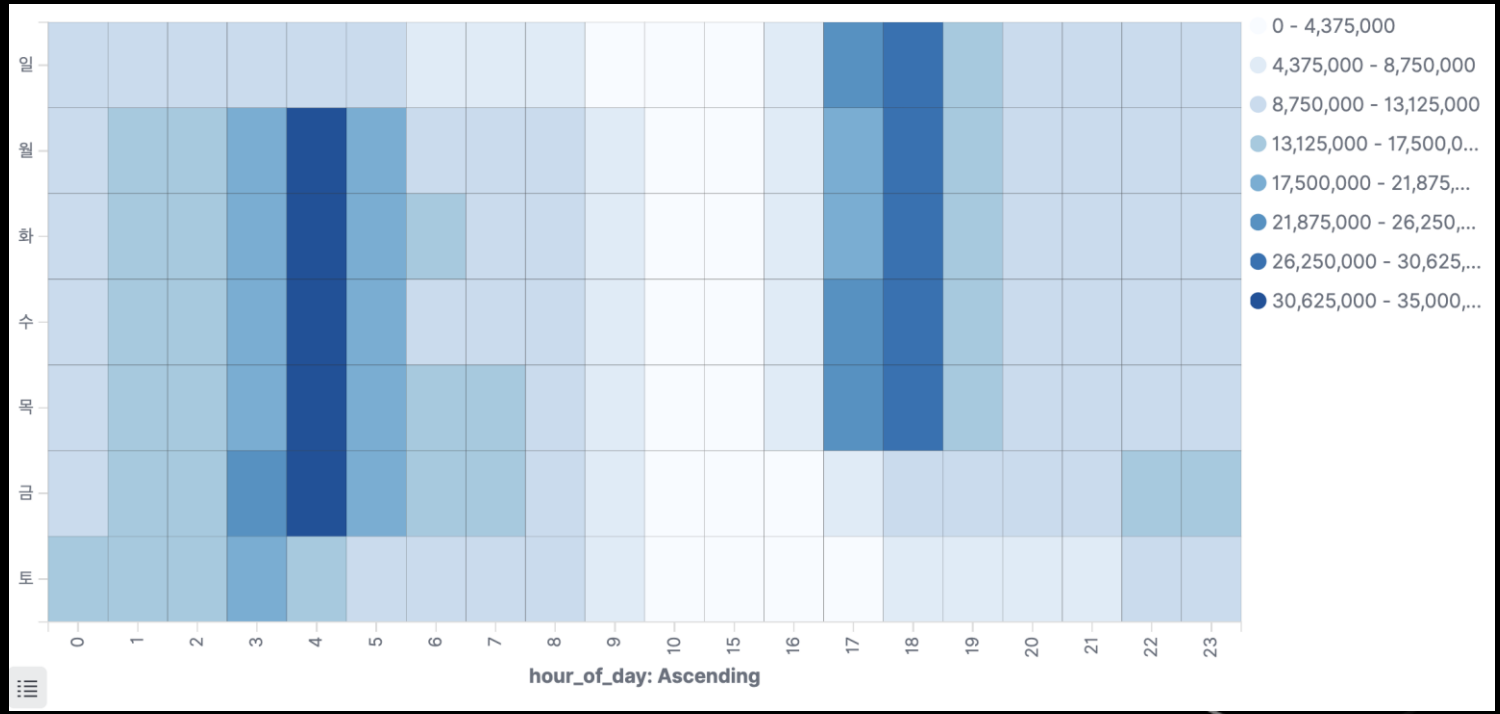


더 추가 해 보고 싶은 데이터

Heat Map : 요일별 시간대별 승하차인원수

- 요일
- 시각

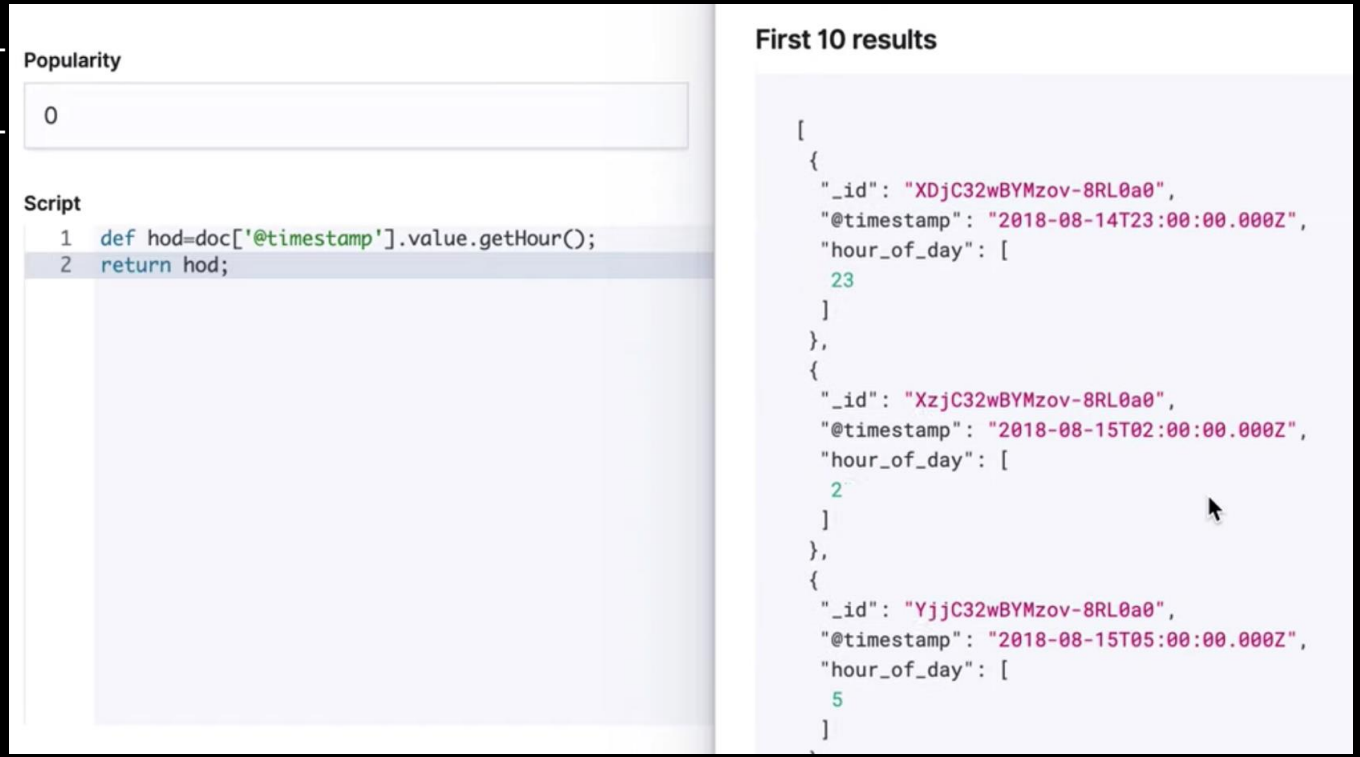
@timestamp?



@timestamp 로 부터 요일, 시각 정보 추가

1. Scripted Field 사용

- 동적으로 추가 가능
- 메모리 소모가 많음



@timestamp 로 부터 요일, 시각 정보 추가

2. 데이터 색인 시 요일, 시각 필드 별도로 추가

- 재색인 필요
- 가벼움
- 더욱 다양한 쿼리 가능

```
GET _ingest/pipeline/hour_and_week
```

```
{
  "hour_and_week" : {
    "description" : "Add hour_of_day and day_of_week field from
      @timestamp",
    "processors" : [
      {
        "script" : {
          "lang" : "painless",
          "source" : """
def ts=ctx['@timestamp'];
def sdf=new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS");
def date=sdf.parse(ts);
def cal=Calendar.getInstance();
cal.setTime(date);

ctx.hour_of_day=cal.get(Calendar.HOUR_OF_DAY);

def dowNum=cal.get(Calendar.DAY_OF_WEEK)-1;
def dowEn=["Sun","Mon","Tue","Wed","Thu","Fri","Sat"][dowNum];
def dowKr=["일","월","화","수","목","금","토"][dowNum];

ctx.day_of_week=["num":dowNum, "en":dowEn, "kr":dowKr];
"""
        }
      }
```

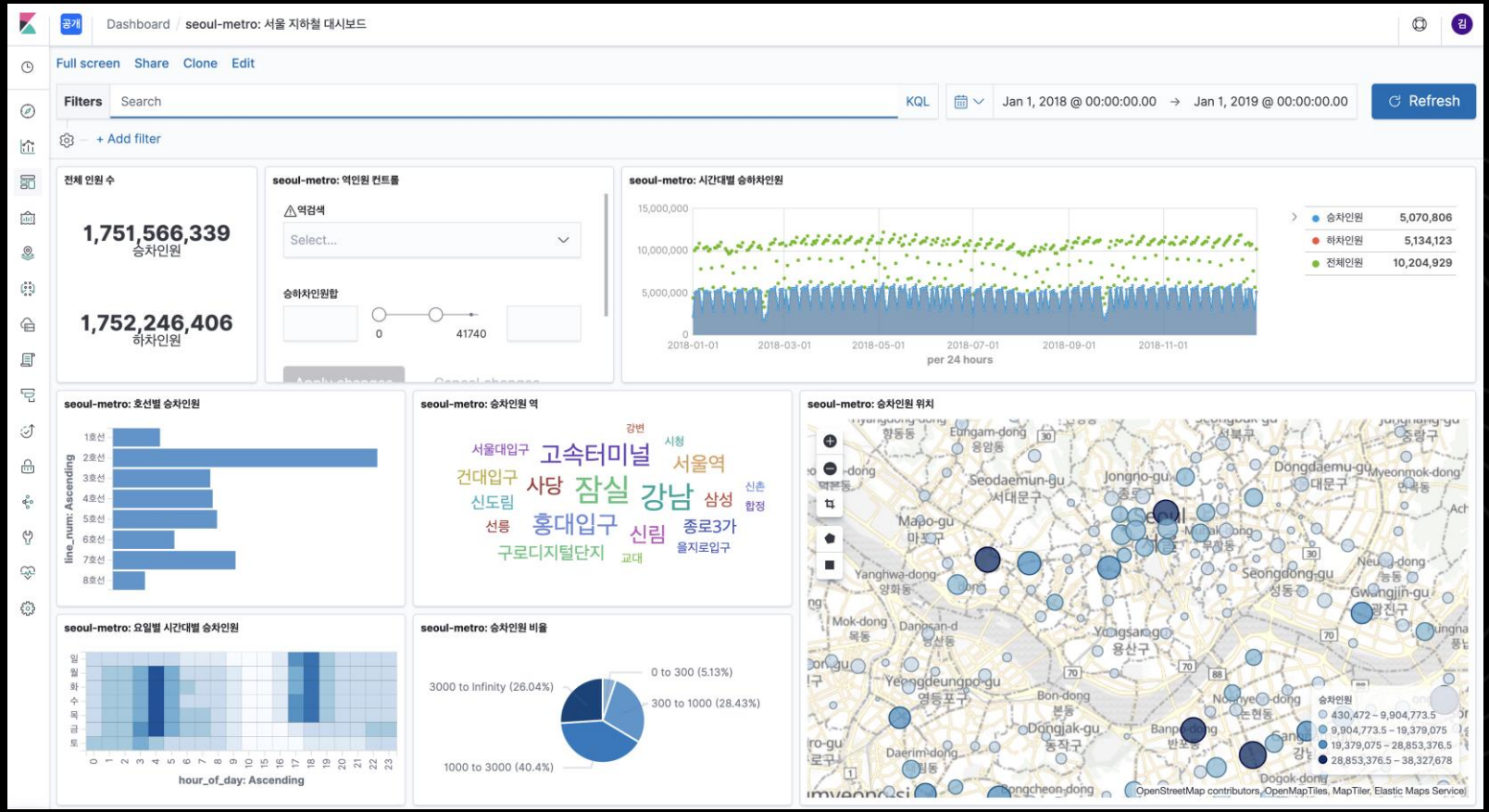
@timestamp 로 부터 요일, 시각 정보 추가

최종 데이터 형태


```
"_source": {
  "code": "2534",
  "line_num": "5호선",
  "people": {
    "total": 14940,
    "in": 14052,
    "out": 888
  },
  "line_num_en": "Line 5",
  "@timestamp": "2018-12-31T10:00:00.000Z",
  "station": {
    "name": "광화문"
  },
  "location": {
    "lon": 126.976669,
    "lat": 37.571026
  },
  "hour_of_day": 10,
  "day_of_week": {
    "num": 1,
    "kr": "월",
    "en": "Mon"
  }
},
```

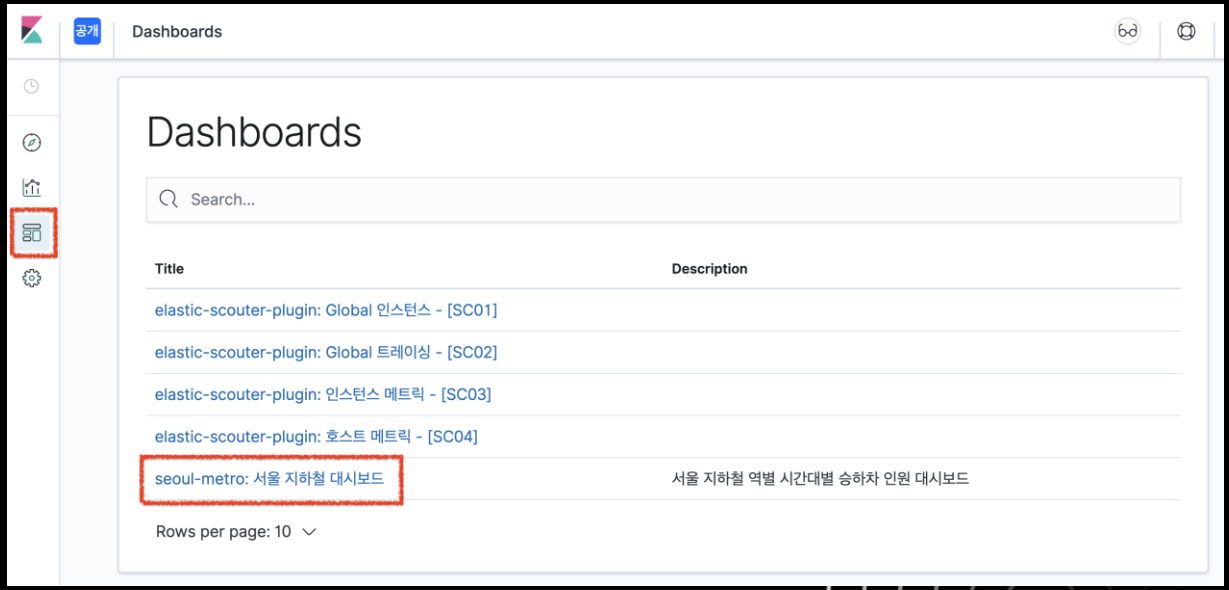

t	code	2534
t	day_of_week.en	Mon
t	day_of_week.kr	월
#	day_of_week.num	1
#	hour_of_day	10
t	line_num	5호선
t	line_num_en	Line 5
	location	{ "lon": 126.976669, "lat": 37.571026 }
#	people.in	14,052
#	people.out	888
#	people.total	14,940
t	station.name	광화문

2차 Kibana 대시보드



실제로 접속해서 보시겠습니다

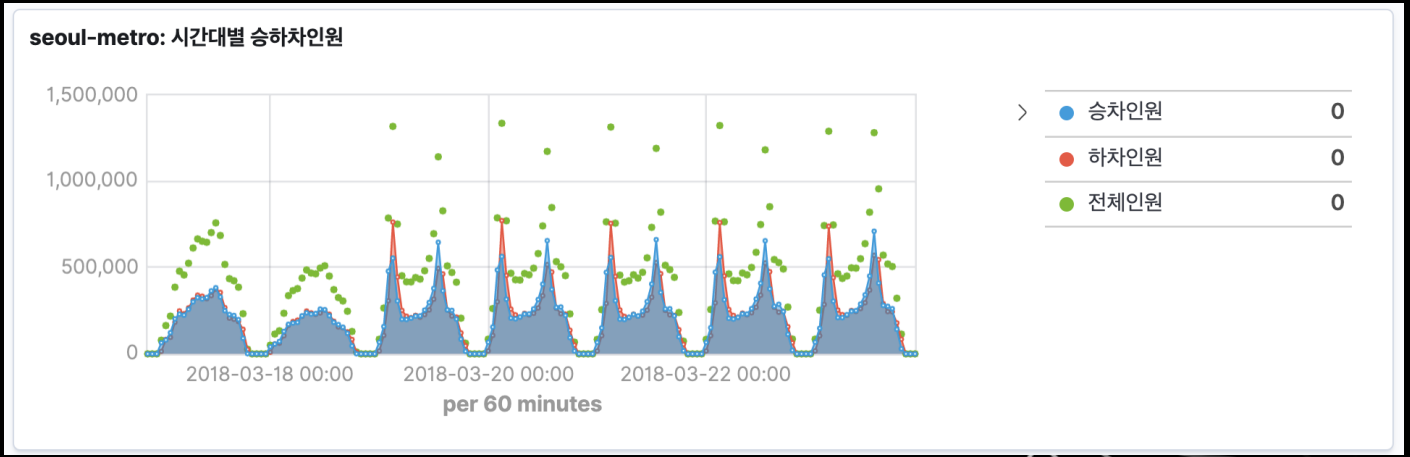
https://ela.st/cloud-kr
Username : guest
Password : thssla (손님)



대시보드 분석

1주일 단위 집계

- 평일 : 아침 저녁
- 주말 : 낮



대시보드 분석

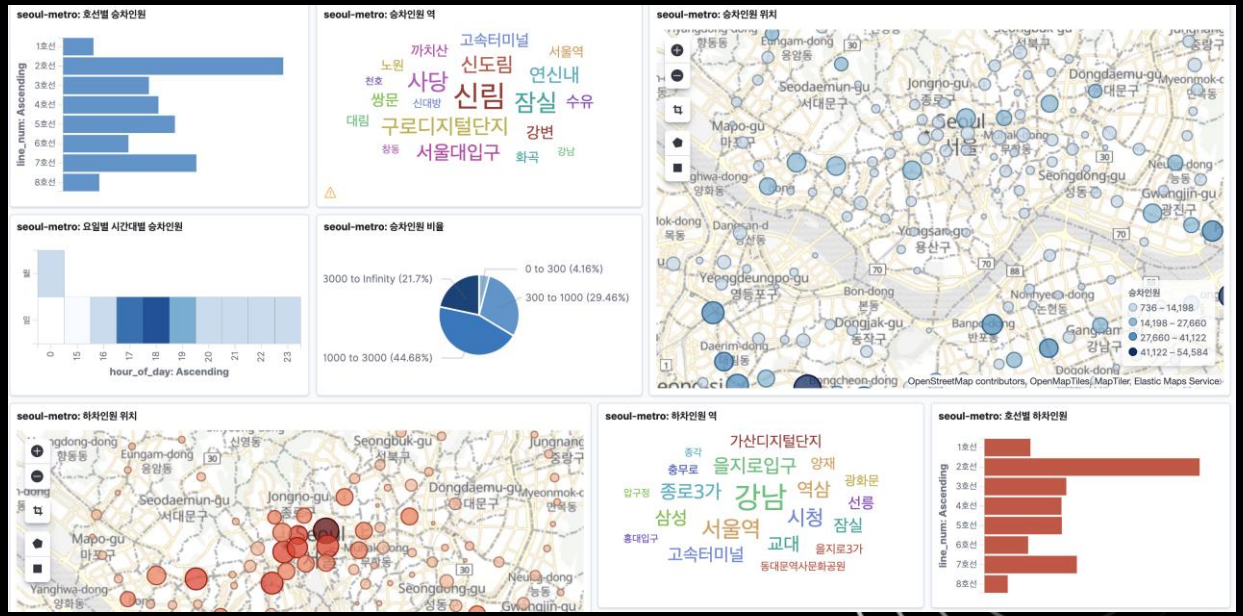
평일 오전

- 승차인원 :

  신림, 잠실, 사당...

- 하차인원 :

  강남, 종로3가, 서울역...



대시보드 분석

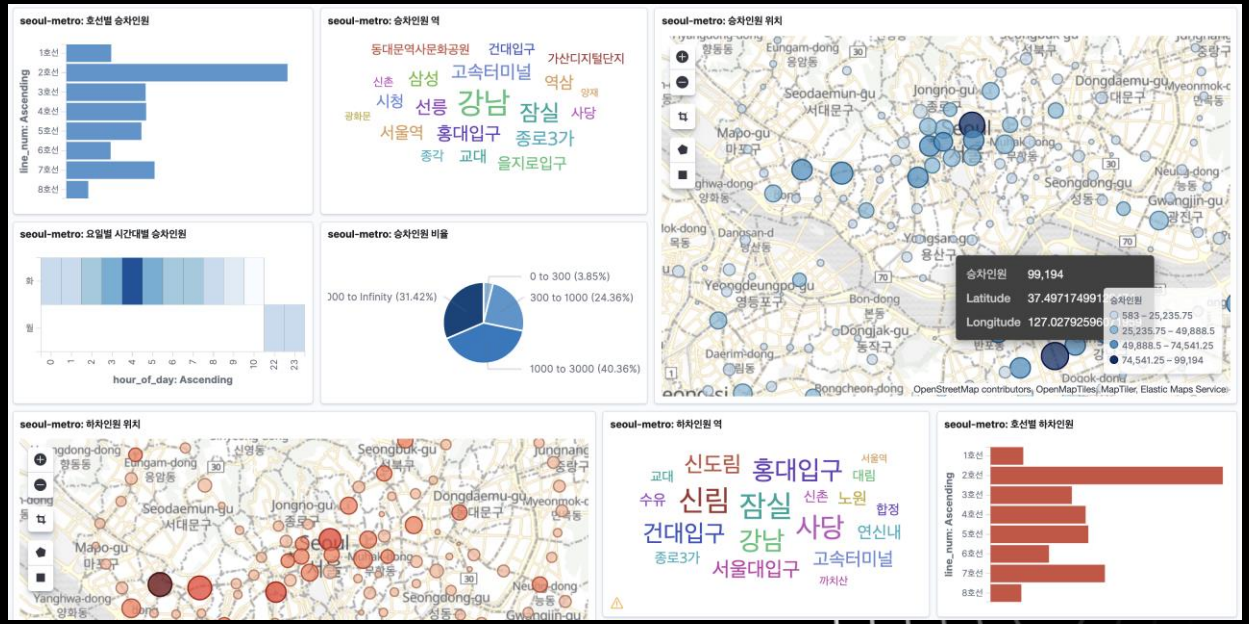
평일 오후

- 승차인원 :

  강남, 잠실, 종로3가...

- 하차인원 :

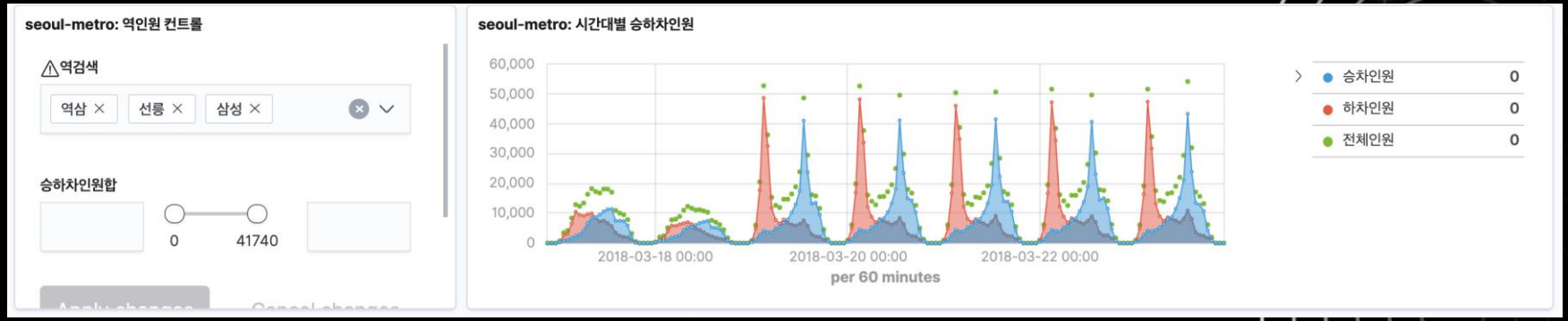
  잠실, 신림, 홍대입구...



대시보드 분석

역삼, 선릉, 삼성 역 으로 검색

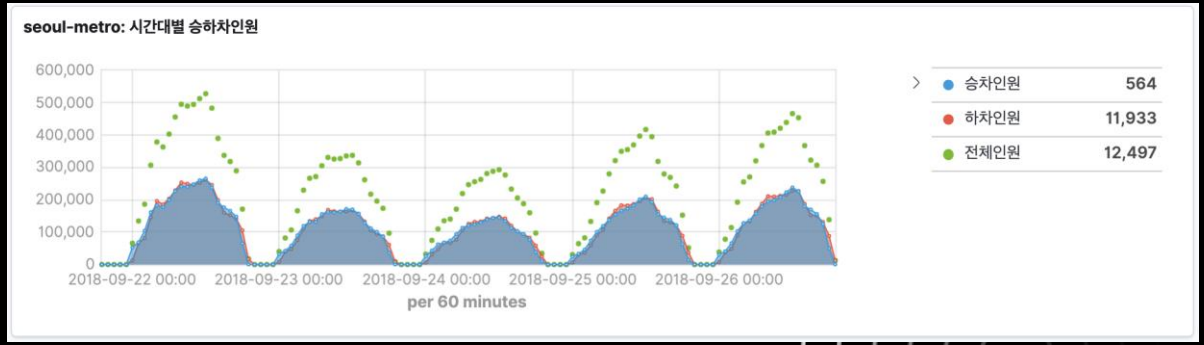
- 오전 : 하차인원
- 오후 : 승차인원

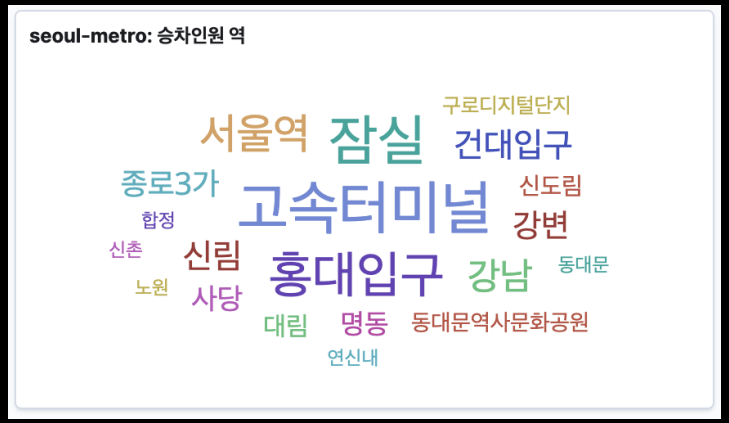


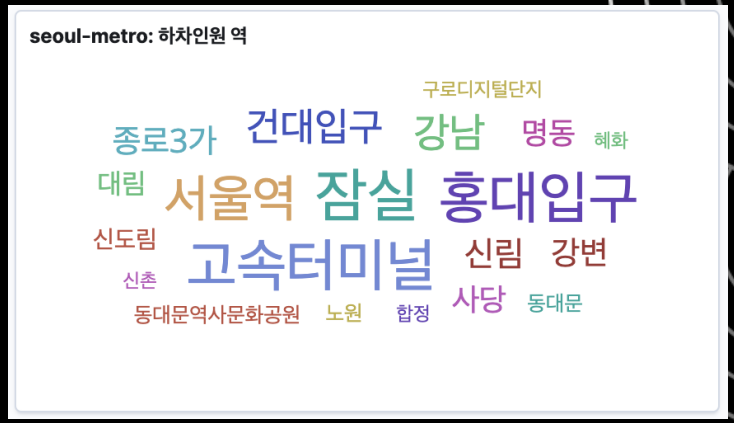
대시보드 분석

명절 기간 검색

- 고속터미널, 홍대입구, 서울역, 잠실

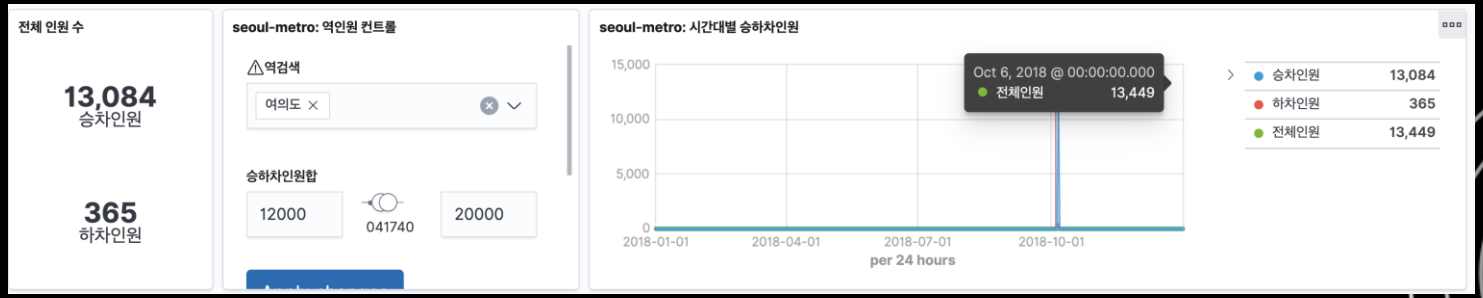


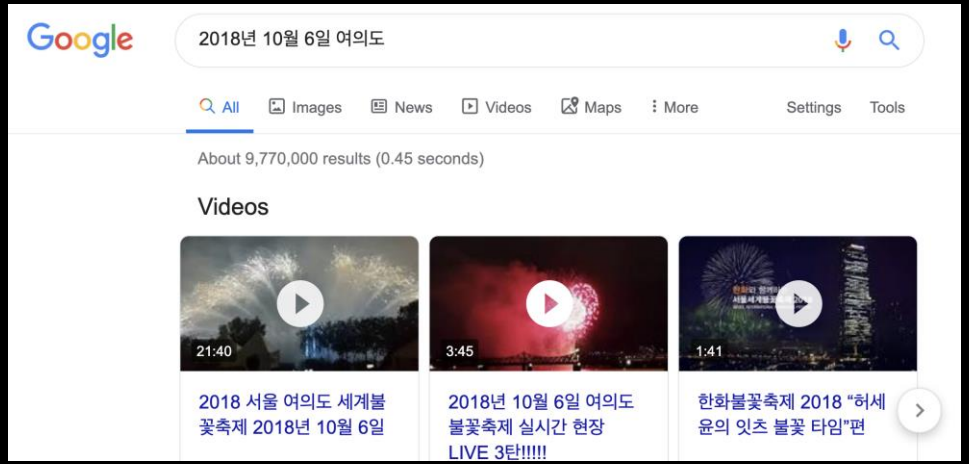




대시보드 분석

여의도에서 승하차 인원 합 : 12,000 ~ 20,000





감사합니다
Thank you

Elastic 부스에서 더 많은 데모와 정보들을 제공하고 있습니다. 추가적인 질문은 Elastic 부스에 오셔서 해 주시면 감사드리겠습니다.