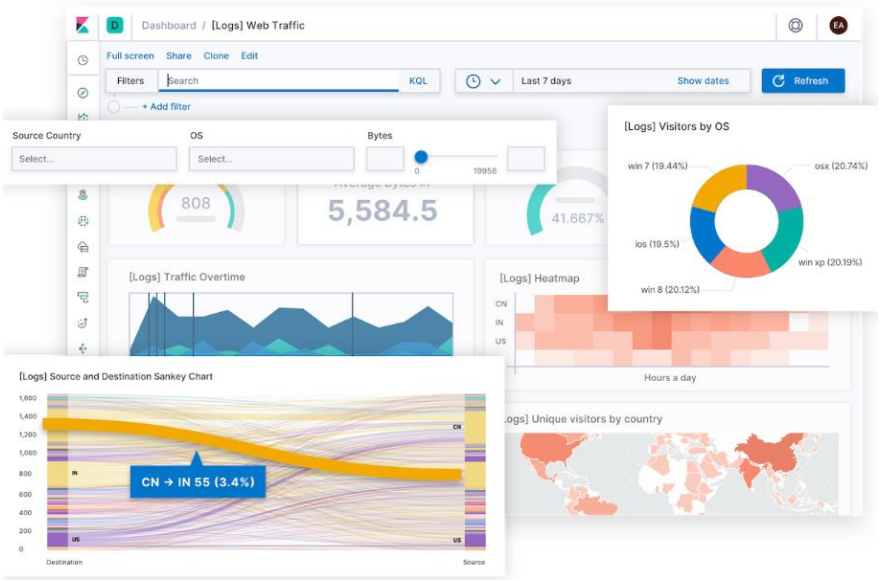
Aggregation - 데이터를 요약하고 분석하는 강력한 도구



목적: 위의 이미지와 같이 다양한 차트들을 표현하는데 쓰이는 유용한 기능

사용방법: **_search API** 에서 query 문과 같은 수준에 지정자 `aggregations` 또는 `aggs`를 명시하고 그 아래 임의의 aggregation 이름을 입력한 뒤 사용할 aggregation 종류와 옵션들을 명시합니다. 한 번의 쿼리로 aggregation 여러 개를 입력가능

기본 구조:

```
GET /<index>/_search
{
  "size": 0,  // 검색 결과는 제외하고 집계 결과만 보기 위해 size를 0으로 설정
  "aggs": {
    "<aggregation_name>": {
      "<aggregation_type>": {
        "field": "<field_name>"
      }
    }
  }
}
```

- **size**: 결과로 반환할 문서의 개수(집계만 원할 경우 `0`으로 설정).
- **aggs**: 집계를 정의하는 필드.
- **aggregation_name**: 집계의 이름(사용자가 정의).
- **aggregation_type**: 집계 유형(예: `terms`, `avg`, `sum`, `date_histogram` 등의 집계를 사용하여 데이터를 그룹화하고 통계를 계산).
- **field_name**: 집계할 필드

실제 예시:

```
GET /filebeat-*/_search
{
  "size": 0,
  "aggs": {
    "status_counts": {
      "terms": {
        "field": "status.keyword",
        "size": 10
      },
      "aggs": {
        "average_duration": {
          "avg": {
            "field": "event.duration"
          }
        }
      }
    }
  }
```

Aggregation의 두 종류: Metrics / Bucket

Metrics aggregation: 숫자 또는 날짜 필드의 값을 가지고 계산

Bucket aggregation: 범위나 keyword 값 등을 가지고 도큐먼트들을 그룹화

Metrics Aggregations

- 가장 흔하게 사용되는 metrics aggregations 은 min, max, sum, avg.

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "all_passangers": {
      "sum": {
        "field": "passangers"
      }
    }
  }
}
```

my_stations 인덱스의 passangers 필드 합 (sum)


```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "all_passangers" : {
      "value" : 41995.0
    }
  }
```

Stats

- min, max, sum, avg 값을 모두 가져와야 한다면 다음과 같이 stats aggregation을 사용

stats 로 passangers 필드의 min, max, sum, a

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "passangers_stats": {
      "stats": {
        "field": "passangers"
      }
    }
  }
}
```


```
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "passangers_stats" : {
      "count" : 10,
      "min" : 971.0,
      "max" : 6478.0,
      "avg" : 4199.5,
      "sum" : 41995.0
    }
  }
}
```


cardinality

- 필드의 값이 모두 몇 종류인지 분포값 계산, 숫. 자 필드나 keyword, ip 필드 등에 사용이 가능.
- 사용자 접속 로그에서 IP 주소 필드를 가지고 실제로 접속한 사용자가 몇명인지 파악하는 등의 용도로 주로 사용

line 필드의 값이 몇 종류인지를 가져오는 aggs

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "uniq_lines ": {
      "cardinality": {
        "field": "line.keyword"
      }
    }
  }
}
```

line 필드의 값이 몇 종류인지를 가져오


```
{
  "took" : 15,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "uniq_lines " : {
      "value" : 3
    }
  }
}
```


위 쿼리 결과 `"uniq_lines " : { "value" : 3 }` 처럼 실제로 line 필드에는 "1호선", "2호선", "3호선" 총 3 종류의 값들을 확인

percentiles, percentile_ranks

- 값들을 백분위 별로 보기위해서 **percentiles** aggregation 의 사용이 가능

passangers 필드의 백분위를 가져오는 aggs

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "pass_percentiles": {
      "percentiles": {
        "field": "passangers"
      }
    }
  }
}
```


```
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "pass_percentiles" : {
      "values" : {
        "1.0" : 971.0000000000001,
        "5.0" : 971.0,
        "25.0" : 2314.0,
        "50.0" : 4766.5,
        "75.0" : 5821.0,
        "95.0" : 6478.0,
        "99.0" : 6478.0
      }
    }
  }
}
```


Bucket Aggregations

- 주어진 조건으로 분류된 **버킷** 들을 만들고, 각 버킷에 소속되는 도큐먼트들을 모아 **그룹으로 구분**

- 각 버킷 별로 포함되는 도큐먼트의 개수는 **doc_count** 값에 기본적으로 표시가 되며 각 버킷 안에 metrics aggregation 을 이용해서 다른 계산들도 가능

Range

- 숫자 필드 값으로 범위를 지정하고 각 범위에 해당하는 버킷을 생성

- **field** 옵션에 해당 필드의 이름을 지정하고 **ranges** 옵션에 배열로 **from**, **to** 값을 가진 오브젝트 값을 나열해서 범위를 지정

* 다음은 passangers 값이 각각 1000 미만, 1000~4000 사이, 4000 이상 인 버킷들을 생성하는 예제

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "passangers_range": {
      "range": {
        "field": "passangers",
        "ranges": [
          {
            "to": 1000
          },
          {
            "from": 1000,
            "to": 4000
          },
          {
            "from": 4000
          }
        ]
      }
    }
  }
}
```

```
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "passangers_range" : {
      "buckets" : [
        {
          "key" : "*-1000.0",
          "to" : 1000.0,
          "doc_count" : 1
        },
        {
          "key" : "1000.0-4000.0",
          "from" : 1000.0,
          "to" : 4000.0,
          "doc_count" : 3
        },
        {
          "key" : "4000.0-*",
          "from" : 4000.0,
          "doc_count" : 6
        }
```

Histogram

- histogram 도 range 와 마찬가지로 숫자 필드의 범위를 나누는 aggs
- histogram 은 from, to 대신 **interval** 옵션을 이용해서 주어진 간격 크기대로 버킷을 구분

* histogram 은 from, to 대신 **interval** 옵션을 이용해서 주어진 간격 크기대로 버킷을 구분

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "passangers_his": {
      "histogram": {
        "field": "passangers",
        "interval": 2000
      }
    }
  }
}
```

```
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "passangers_his" : {
      "buckets" : [
        {
          "key" : 0.0,
          "doc_count" : 2
        },
        {
          "key" : 2000.0,
          "doc_count" : 2
        },
        {
          "key" : 4000.0,
          "doc_count" : 4
        },
        {
          "key" : 6000.0,
          "doc_count" : 2
        }
      ]
    }
  }
}
```

date_range / date_histogram

- 날짜 필드를 이용해서 범위별로 버킷의 생성이 가능
- **date_range** 는 **ranges** 옵션에 `{"from": "2019-06-01", "to": "2016-07-01"}` 와 같이 입력하며 **date_histogram** 은 **interval** 옵션에 `day, month, week` 와 같은 값들을 이용해서 날짜 간격을 지정

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "date_his": {
      "date_histogram": {
        "field": "date",
        "interval": "month"
      }
    }
  }
}
```

```
"aggregations" : {
  "date_his" : {
    "buckets" : [
      {
        "key_as_string" : "2019-06-01T00:00:00.000Z",
        "key" : 1559347200000,
        "doc_count" : 2
      },
      {
        "key_as_string" : "2019-07-01T00:00:00.000Z",
        "key" : 1561939200000,
        "doc_count" : 2
      },
      {
        "key_as_string" : "2019-08-01T00:00:00.000Z",
        "key" : 1564617600000,
        "doc_count" : 2
      },
      {
        "key_as_string" : "2019-09-01T00:00:00.000Z",
        "key" : 1567296000000,
        "doc_count" : 3
      },
      {
        "key_as_string" : "2019-10-01T00:00:00.000Z",
        "key" : 1569888000000,
        "doc_count" : 1
      }
    ]
```

Terms

- **keyword** 필드의 문자열 별로 버킷을 나누어 집계가 가능
- **keyword** 필드 값으로만 사용이 가능하며 분석된 text 필드는 일반적으로는 사용이 불가능

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "stations": {
      "terms": {
        "field": "station.keyword
      }
    }
  }
}
```

```
"aggregations" : {
  "stations" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "강남",
        "doc_count" : 5
      },
      {
        "key" : "불광",
        "doc_count" : 1
      },
      {
        "key" : "신촌",
        "doc_count" : 1
      },
      {
        "key" : "양재",
        "doc_count" : 1
      },
      {
        "key" : "종각",
        "doc_count" : 1
      },
      {
        "key" : "홍제",
        "doc_count" : 1
      }
    ]
```

sub-aggregations

- Bucket Aggregation 으로 만든 버킷들 내부에 다시 "aggs" : { } 를 선언해서 또다른 버킷을 만들거나 Metrics Aggregation 을 만들어 사용이 가능

* terms aggregation을 이용해서 생성한 **stations** 버킷 별로 **avg** aggregation을 이용해서 **passangers** 필드의 평균값을 계산하는 **avg_psg_per_st** 을 생성하는 예제

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "stations": {
      "terms": {
        "field": "station.keyword"
      },
      "aggs": {
        "avg_psg_per_st": {
          "avg": {
            "field": "passangers"
          }
        }
      }
    }
  }
}
```


```
"aggregations" : {
  "stations" : {
    "doc_count_error_upper_bound" : 0,
    "sum_other_doc_count" : 0,
    "buckets" : [
      {
        "key" : "강남",
        "doc_count" : 5,
        "avg_psg_per_st" : {
          "value" : 5931.2
        }
      },
      {
        "key" : "불광",
        "doc_count" : 1,
        "avg_psg_per_st" : {
          "value" : 971.0
        }
      },
      {
        "key" : "신촌",
        "doc_count" : 1,
        "avg_psg_per_st" : {
          "value" : 3912.0
        }
      },
      {
        "key" : "양재",
        "doc_count" : 1,
        "avg_psg_per_st" : {
          "value" : 4121.0
        }
      },
      {
        "key" : "종각",
        "doc_count" : 1,
        "avg_psg_per_st" : {
          "value" : 2314.0
        }
      },
      {
        "key" : "홍제",
        "doc_count" : 1,
        "avg_psg_per_st" : {
          "value" : 1021.0
        }
      }
```

Pipeline Aggregations

- pipeline 에는 다른 버킷의 결과들을 다시 연산하는 **min_bucket**, **max_bucket**, **avg_bucket**, **sum_bucket**, **stats_bucket**, 이동 평균을 구하는 **moving_avg**, 미분값을 구하는 **derivative**, 값의 누적 합을 구하는 **cumulative_sum** 등이 있음

- `"buckets_path": "<버킷 이름>"` 옵션을 이용해서 입력 값으로 사용할 버킷을 지정

```
GET my_stations/_search
{
  "size": 0,
  "aggs": {
    "months": {
      "date_histogram": {
        "field": "date",
        "interval": "month"
      },
      "aggs": {
        "sum_psg": {
          "sum": {
            "field": "passangers"
          }
        },
        "accum_sum_psg": {
          "cumulative_sum": {
            "buckets_path": "sum_psg"
          }
        }
      }
    }
```


```
},
"aggregations" : {
  "months" : {
    "buckets" : [
      {
        "key_as_string" : "2019-06-01T00:00:00.000Z",
        "key" : 1559347200000,
        "doc_count" : 2,
        "sum_psg" : {
          "value" : 7726.0
        },
        "accum_sum_psg" : {
          "value" : 7726.0
        }
      },
      {
        "key_as_string" : "2019-07-01T00:00:00.000Z",
        "key" : 1561939200000,
        "doc_count" : 2,
        "sum_psg" : {
          "value" : 12699.0
        },
        "accum_sum_psg" : {
          "value" : 20425.0
        }
      },
      {
        "key_as_string" : "2019-08-01T00:00:00.000Z",
        "key" : 1564617600000,
        "doc_count" : 2,
        "sum_psg" : {
          "value" : 11545.0
        },
        "accum_sum_psg" : {
          "value" : 31970.0
        }
      },
      {
        "key_as_string" : "2019-09-01T00:00:00.000Z",
        "key" : 1567296000000,
        "doc_count" : 3,
        "sum_psg" : {
          "value" : 9054.0
        },
        "accum_sum_psg" : {
          "value" : 41024.0
        }
      },
      {
        "key_as_string" : "2019-10-01T00:00:00.000Z",
        "key" : 1569888000000,
        "doc_count" : 1,
        "sum_psg" : {
          "value" : 971.0
        },
        "accum_sum_psg" : {
          "value" : 41995.0
```


`"accum_sum_psg" : { "value" : 7726.0}` = 7726.0

`"accum_sum_psg" : { "value" : 20425.0}` = 7726.0 + 12699.0

`"accum_sum_psg" : { "value" : 31970.0}` = 7726.0 + 12699.0 + 11545.0

...