Elastic Stack을 활용한 Data Dashboard 만들기

Week 2 - Data를 시각화해보자 II

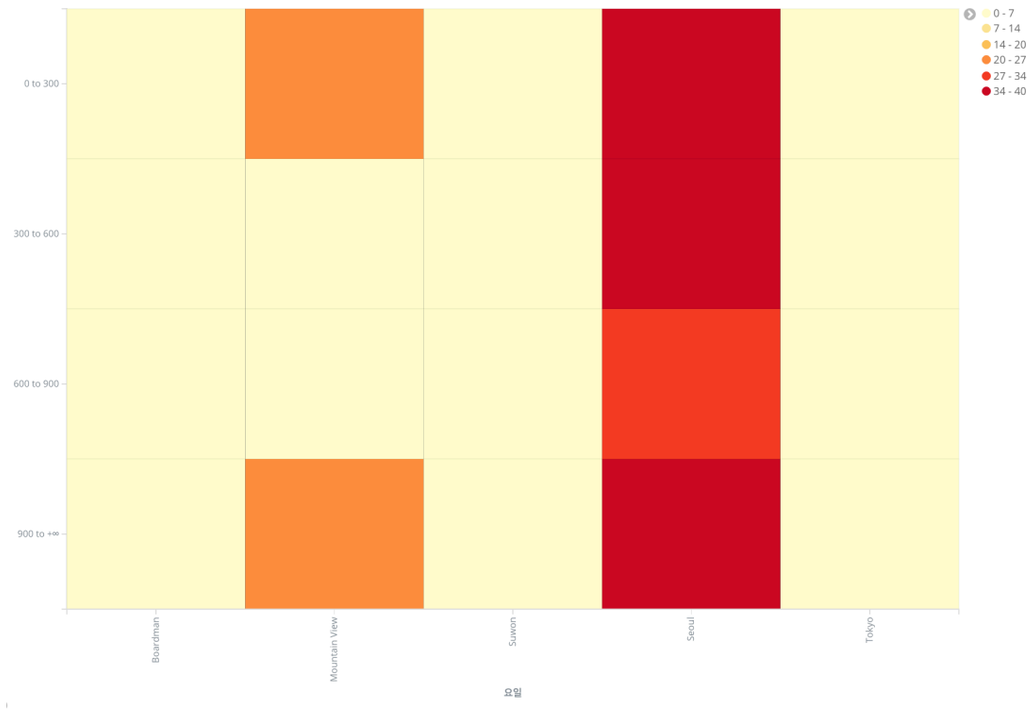
Fast Campus

지난 시간에 못 다룬 Visualization Type를 간단하 살펴보자

Heat Map

- 색을 통해 특정 metrics aggregation 결과 시각화
- bucket은 (Chart 당) 최대 2개 생성 가능 (x축, y축)
- 주로 Categorical Field Data에 적용

Heat Map Object



해석

- **nginx-*** index의
- “@timestamp” field 기준 “**2018-06-01 ~ 2018-08-12**” documents의
- “**nginx.access.body_sent.bytes**” field 값의 **평균**이 **가장 큰**
- “**nginx.access.geoip.city_name**” field **5개** 별
- "**nginx.access.body_sent.bytes**” field로 아래와 같이 bucket을 생성한 후
  - 0 ~ 300
  - 300 ~ 600
  - 600 ~ 900
  - 900 ~
- “**nginx.access.remote_ip**” field의 **unique 개수**

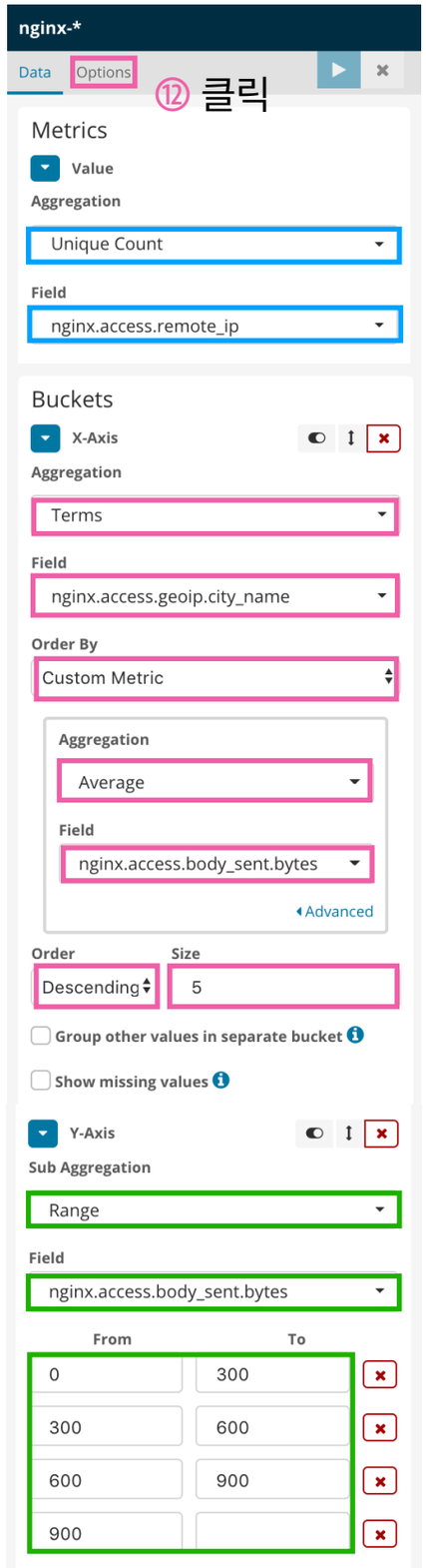
(x축) 도시별

(y축) 데이터 크기별

unique한 ip 주소 개수

5

Heat Map Configuration 1



① Unique Count aggregation 선택

② Unique Count aggregation 적용할 Field 선택

② Terms aggregation 선택

③ Terms aggregation 적용할 Field 선택

④ Terms aggregation 후 bucket 선정 기준 선택

⑤ Custom Metric 선택

⑥ Custom Metric 적용할 Field 선택

⑦ 정렬방식 (오름/내림) 선택

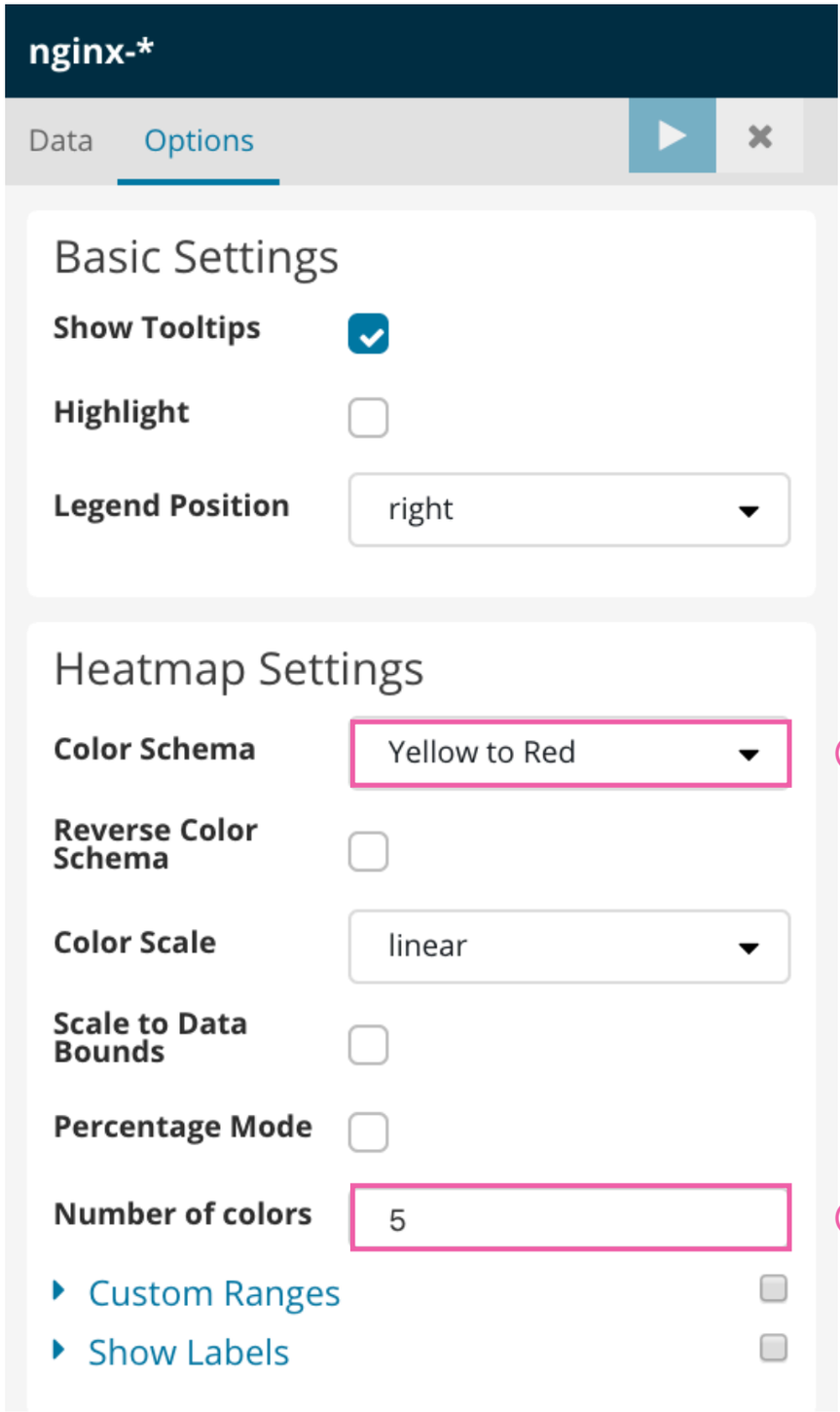
⑧ 반영할 bucket 개수 입력

⑨ Range aggregation 선택

⑩ Range aggregation 적용할 Field 선택

⑪ bucket 별 Range 직접 입력

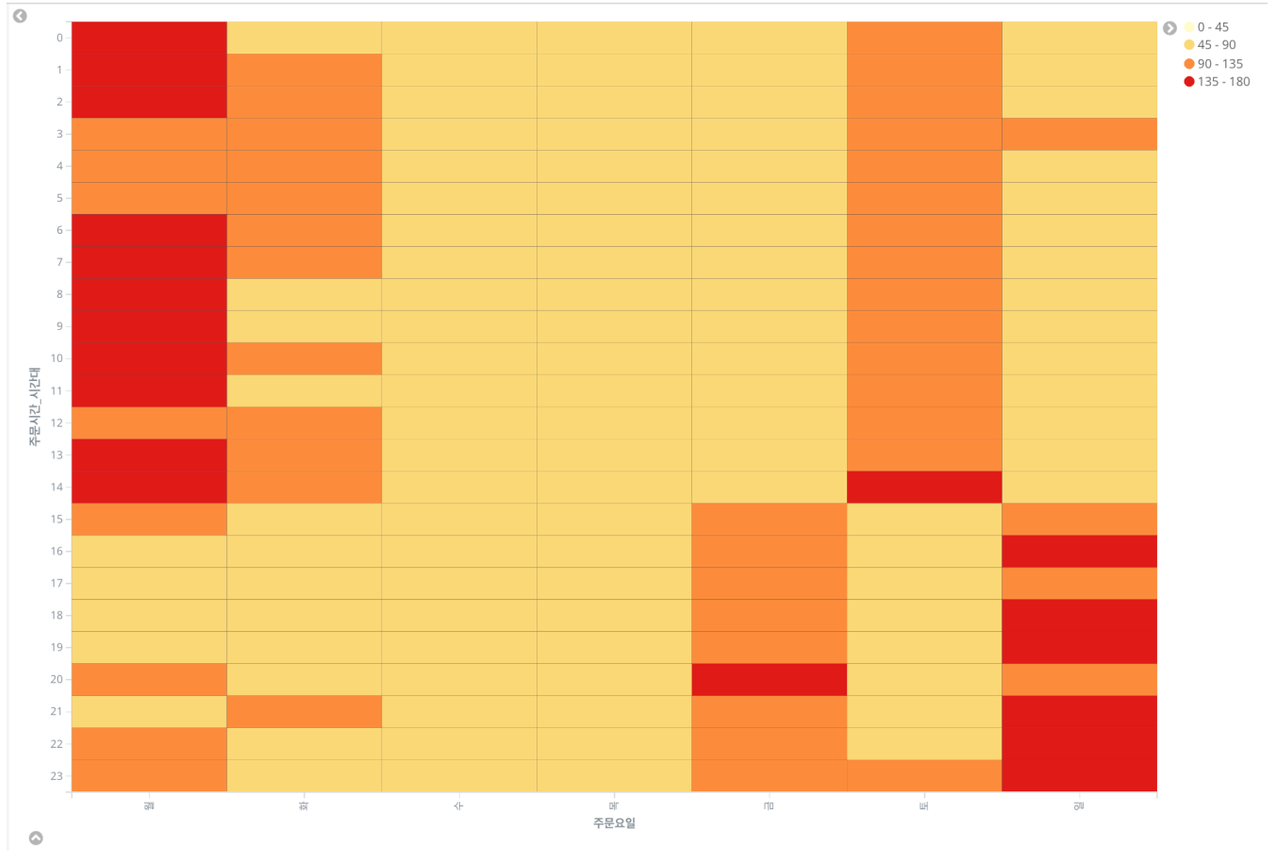
Heat Map Configuration 2



⑬ 적절한 Color Schema 선택

⑭ 사용할 Color 개수 입력 (세분화)

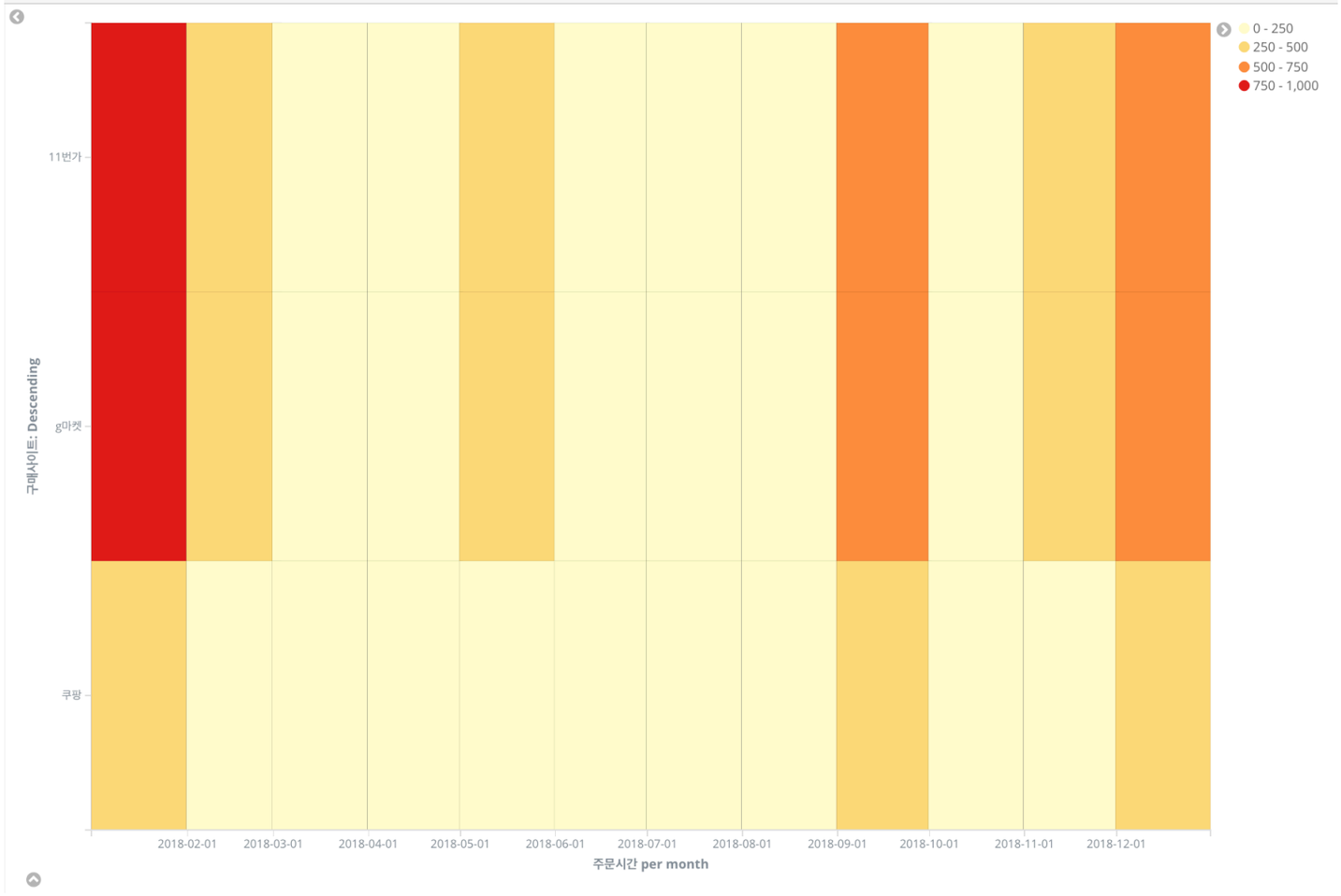
예제 8) Heat Map



조건

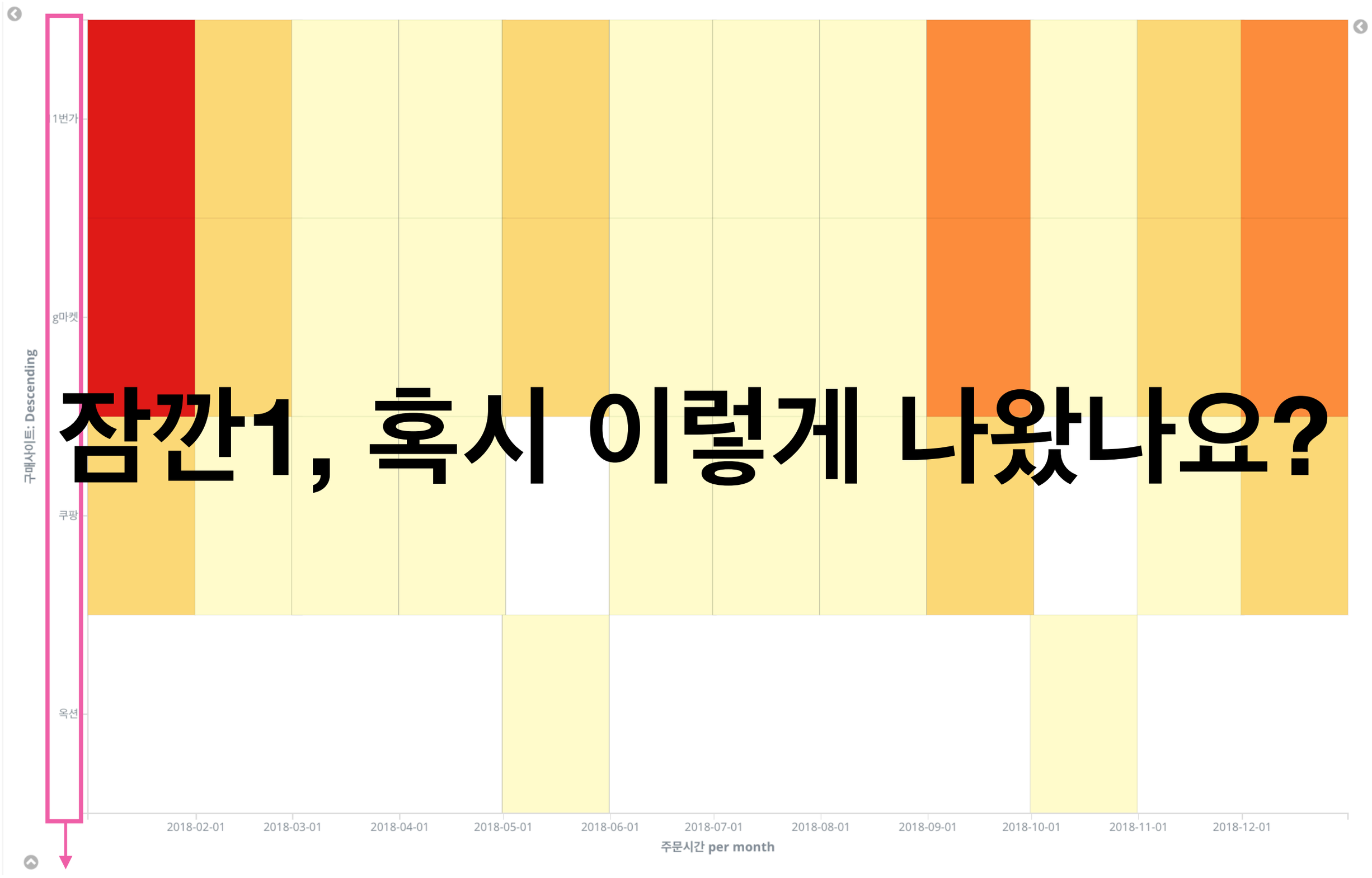
- **shopping** index 중에서
- “주문시간” field 기준 **this year** documents를
- “**주문시간_요일_sort**“ field 값의 **평균이 큰 7개**의 “**주문시간_요일**” field 별 —— **(x축) 일별**
- 1 단위로 나눈 “**주문시간_시간대**” field 별 —— **(y축) 시간대별**
- document **개수** —— **판매수**

예제 9) Heat Map



조건

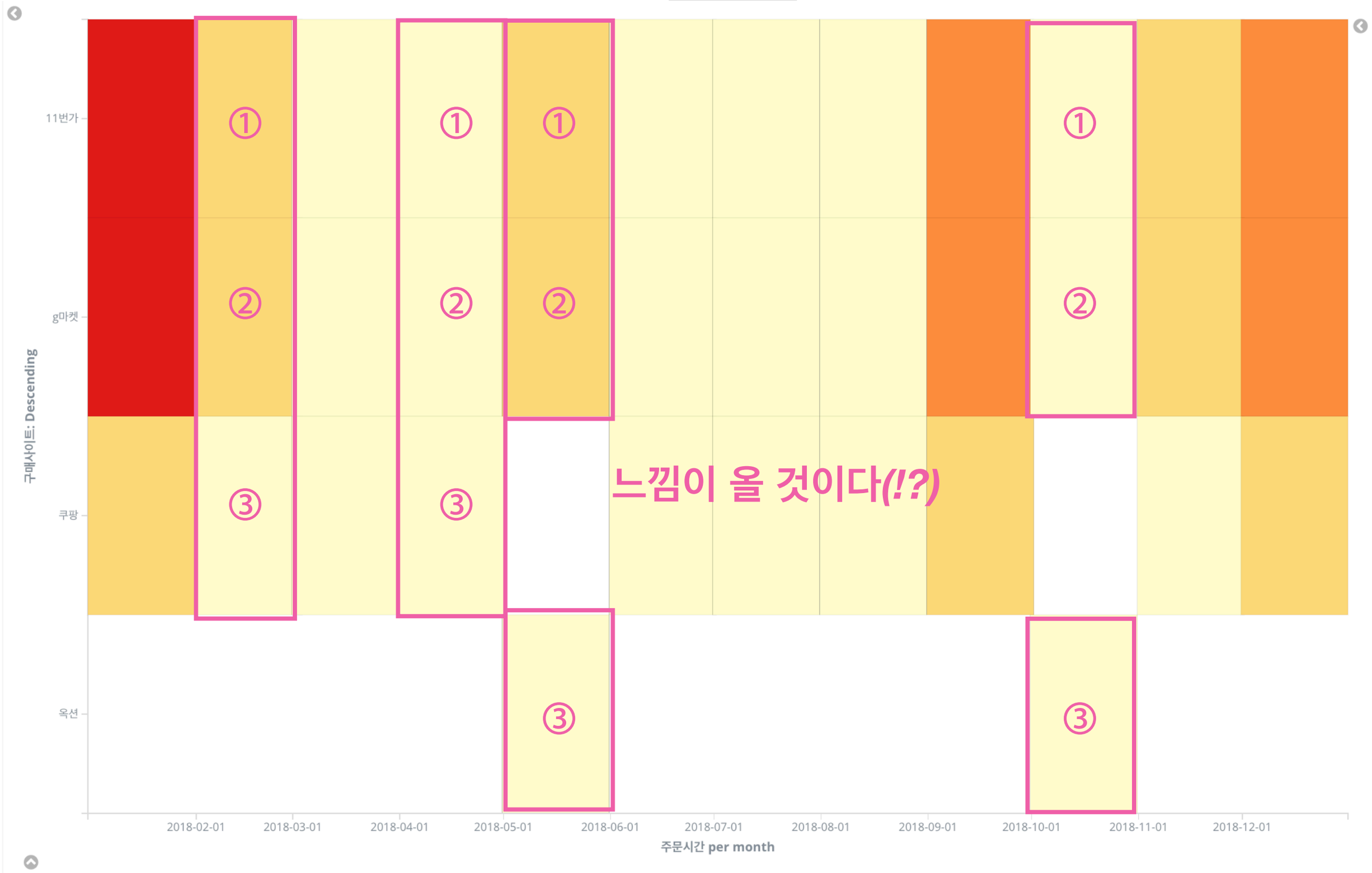
- **shopping** index 중에서
- “주문시간” field 기준 **this year** documents를
- documents **개수**가 가장 많았던 **3개**의 “**구매사이트**” field 별로 —— **(y축) 구매사이트 별**
- “**주문시간**” field를 기준으로 **월별**(monthly) bucket을 생성한 후 —— **(x축) 월별**
- documents **개수** —— **판매수**



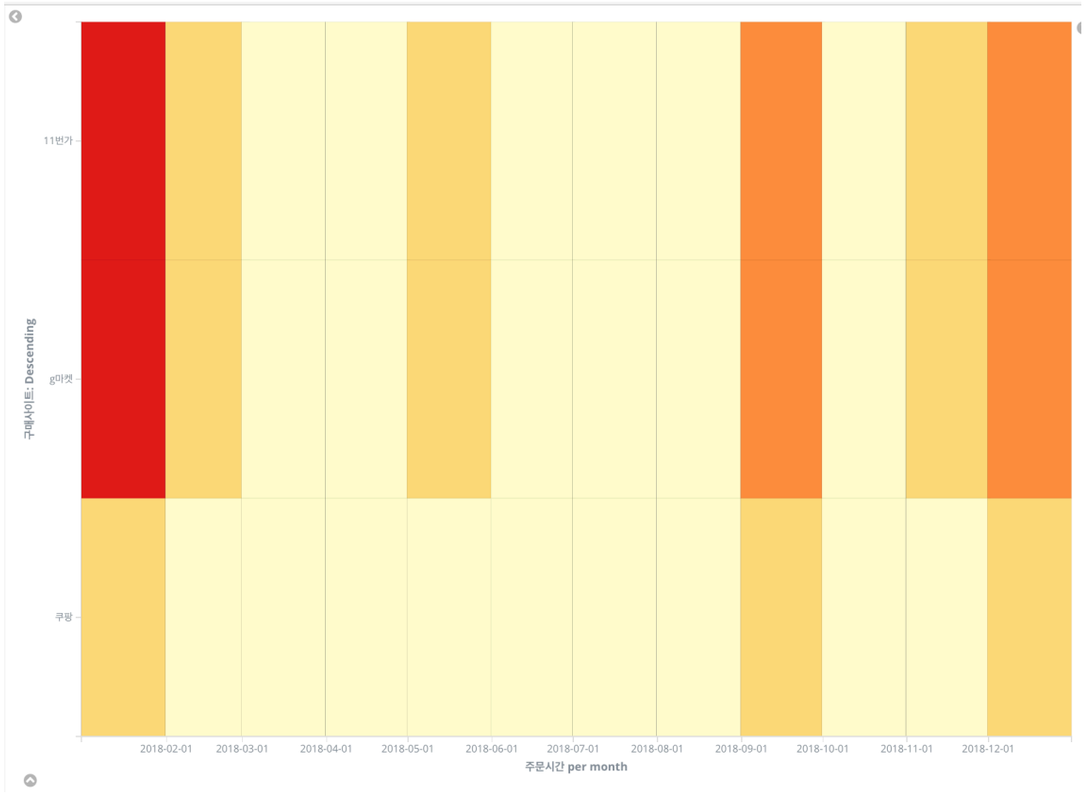
잠깐1, 혹시 이렇게 나왔나요?

왜 4개가 나올까?

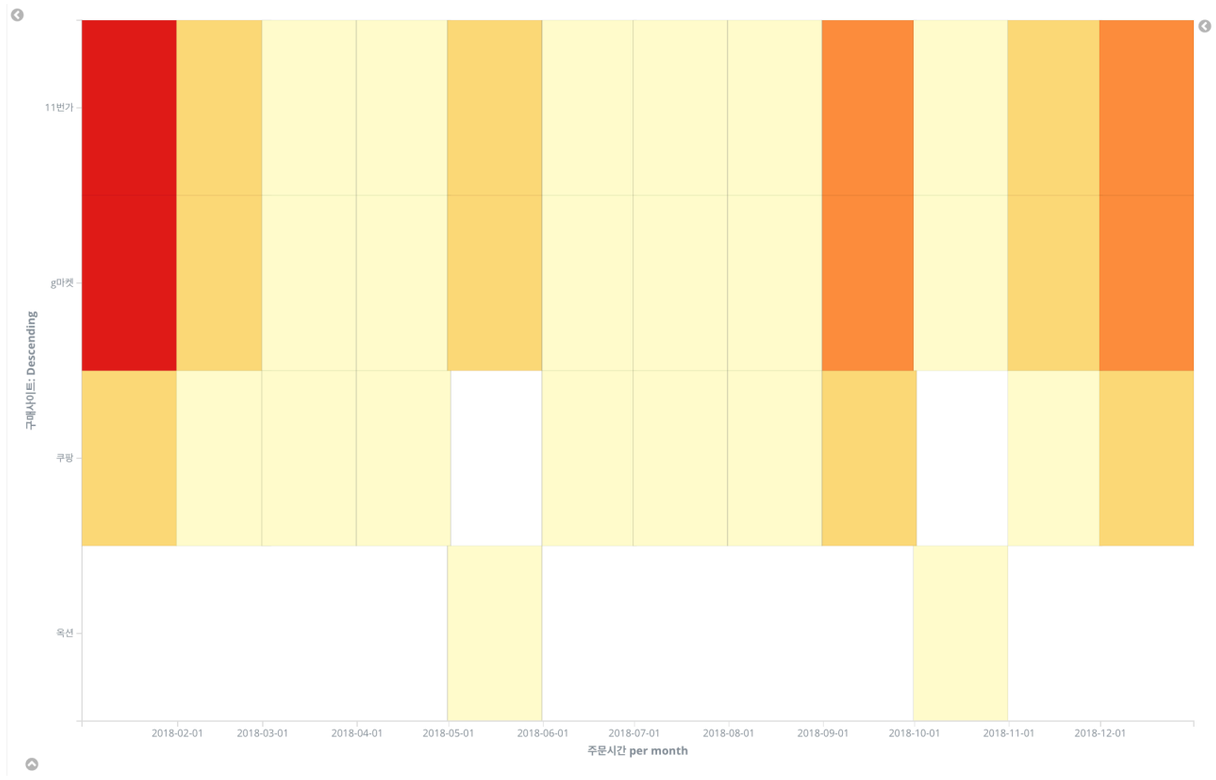
예제9) 힌트


11번가
g마켓
쿠팡
옥션
①
②
③
느낌이 올 것이다(!?)
2018-02-01
2018-03-01
2018-04-01
2018-05-01
2018-06-01
2018-07-01
2018-08-01
2018-09-01
2018-10-01
2018-11-01
2018-12-01
주문시간 per month
구매수량: Descending

Bucket Aggregation 순서 주의



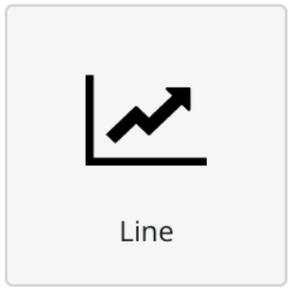
vs



documents 개수가 많았던 구매사이트 3개에 대해서 월별로....

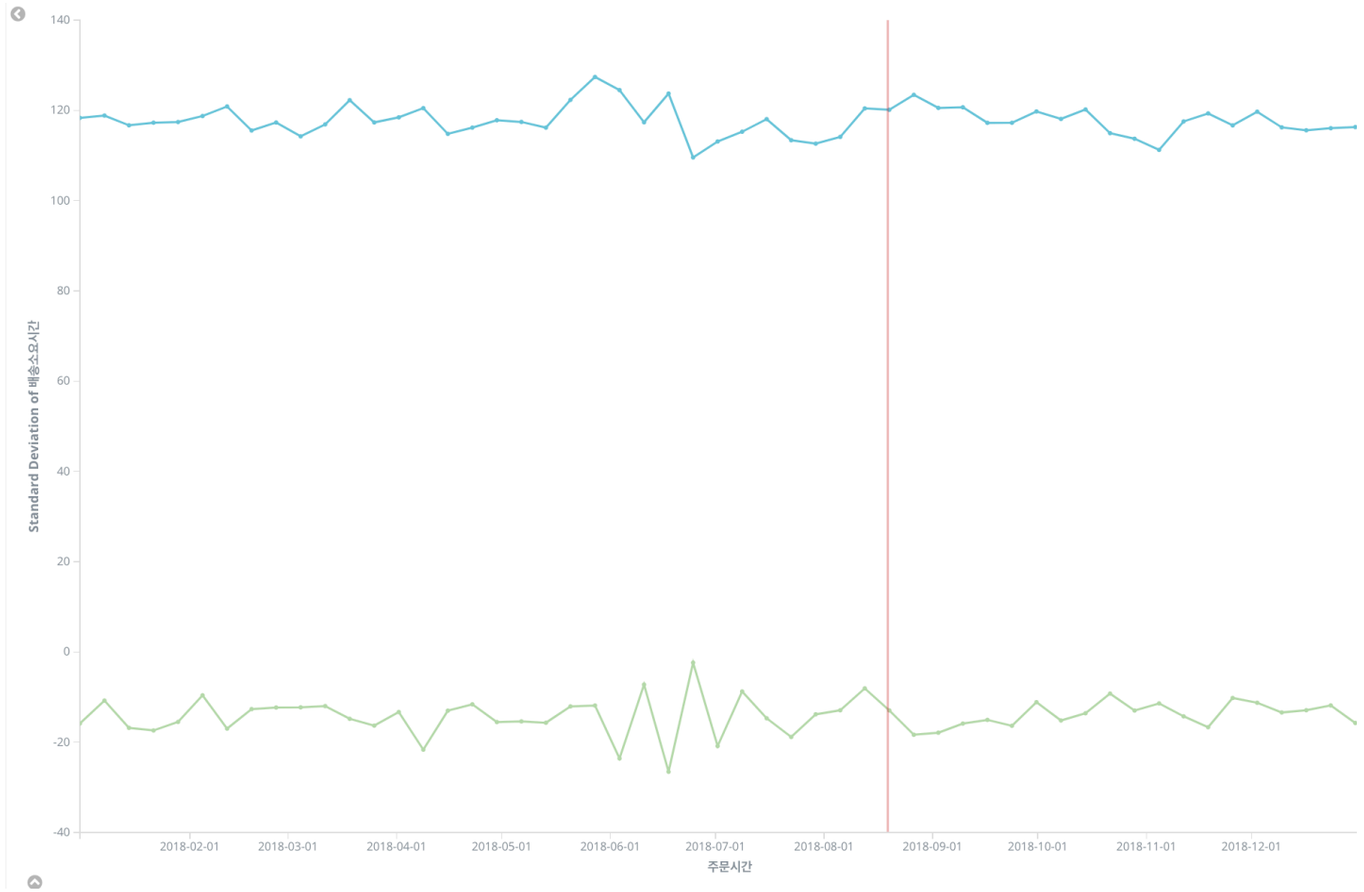
월별로 documents 개수가 많았던 구매사이트 3개에 대해서...

Line Chart



- x축의 변화에 따른 특정 field의 값의 추이를 볼 때 사용
- 주로 x축은 날짜/시간 관련 field를 사용
- kibana 설정은 line // bar // area 모두 유사

Line Chart Object

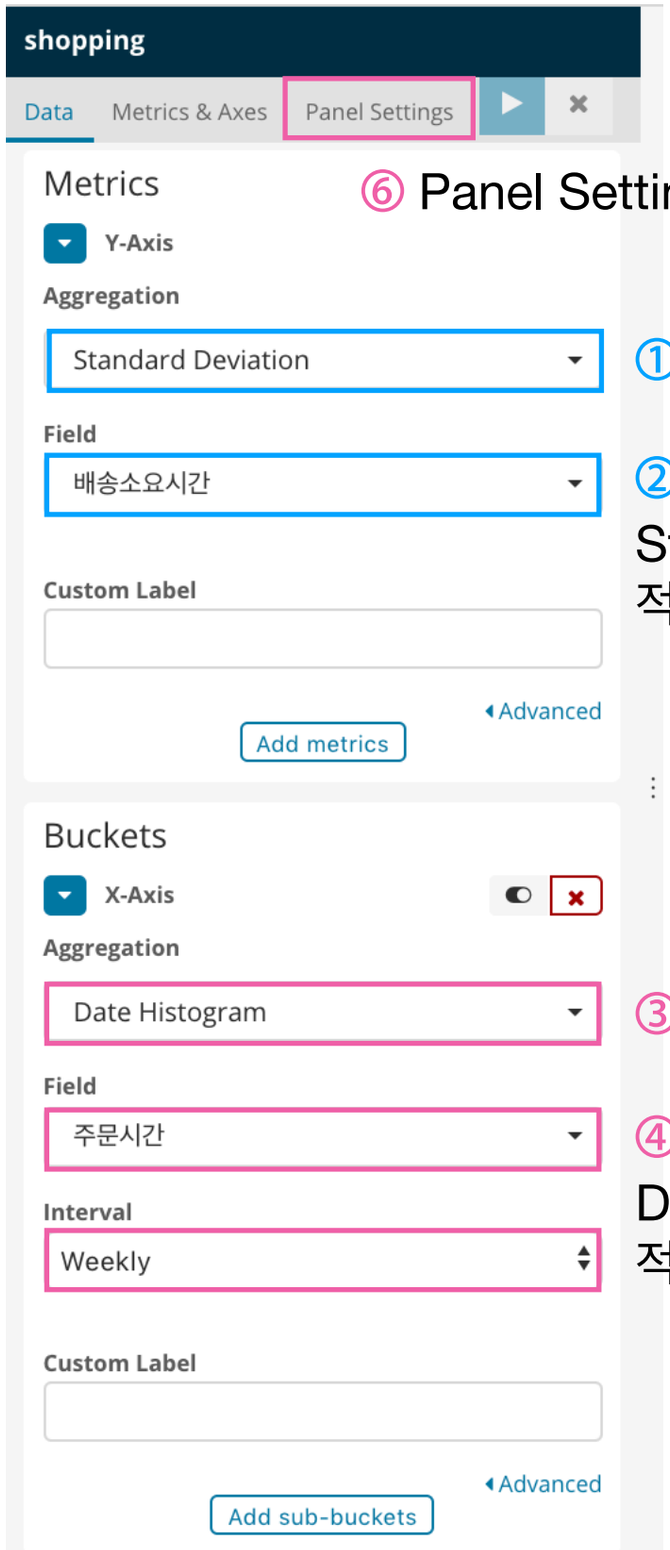


해석

- **shopping** index의
- “주문시간” field 기준 **this year** documents의
- “**주문시간**” field를 기준으로 **주별**(weekly) bucket을 생성한 후 —— **주별**
- “**배송소요시간**” field의 **표준편차** —— **배송소요시간의 표준편차**

14

Line Chart Configuration 1



⑥ Panel Settings 선택

① Standard Deviation Aggregation 선택

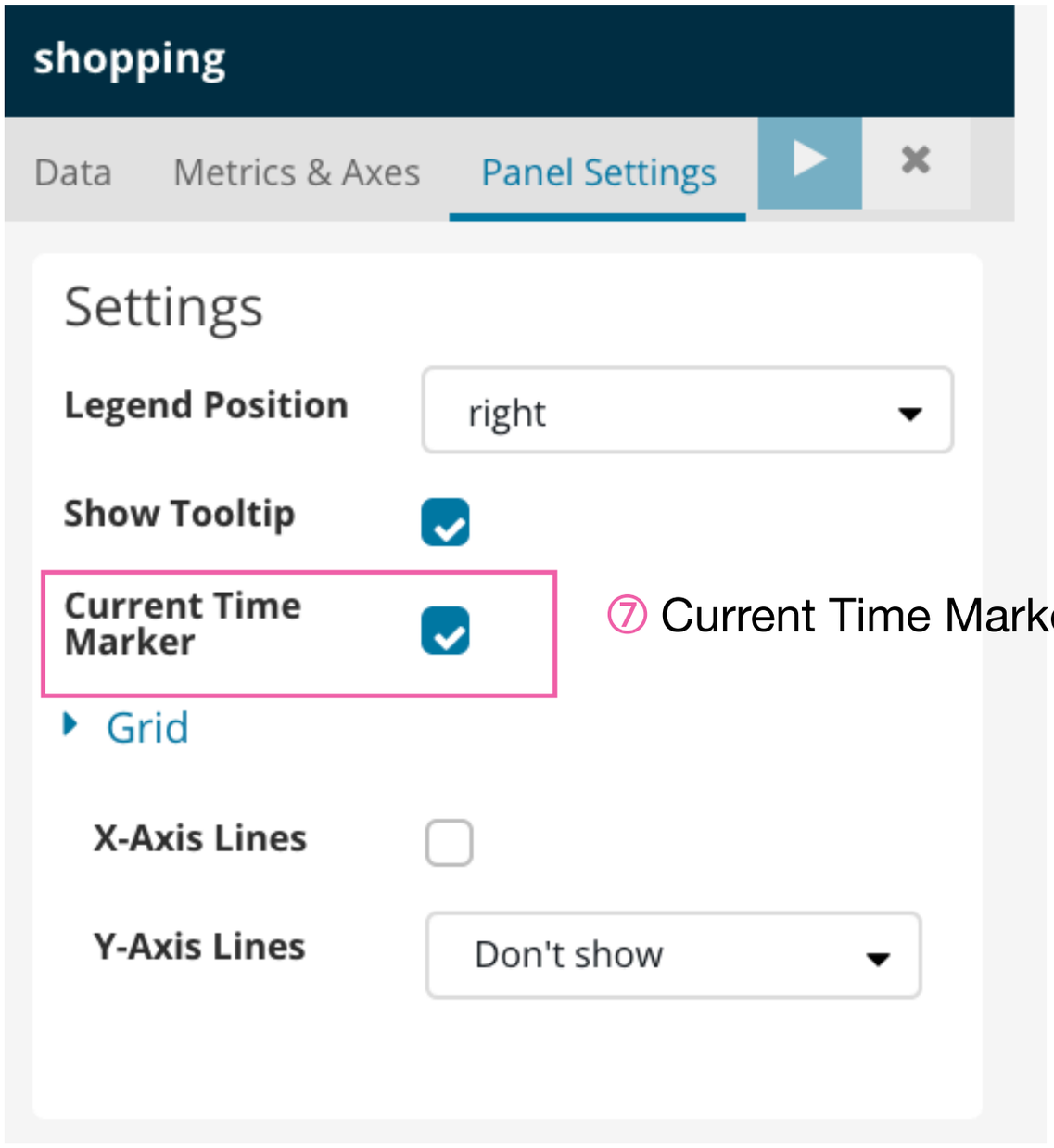
②
Standard Deviation Aggregation
적용할 Field 선택

③ Date Histogram Aggregation 선택

④
Date Histogram Aggregation
적용할 Field 선택

⑤ Date Histogram 간격 설정

Line Chart Configuration 2

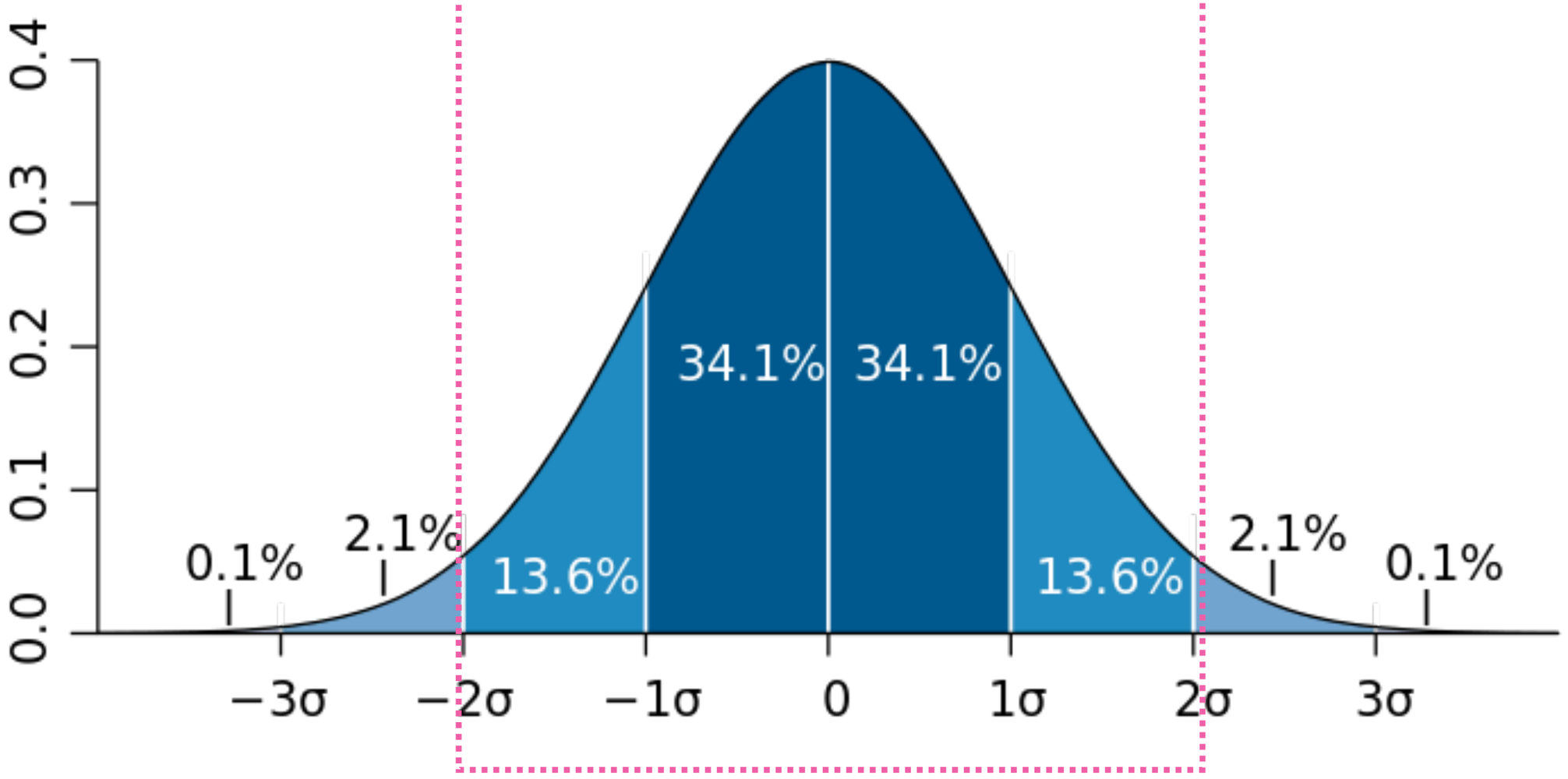


⑦ Current Time Marker 선택

잠깐2, 왜 표준편차 값이 2개지?

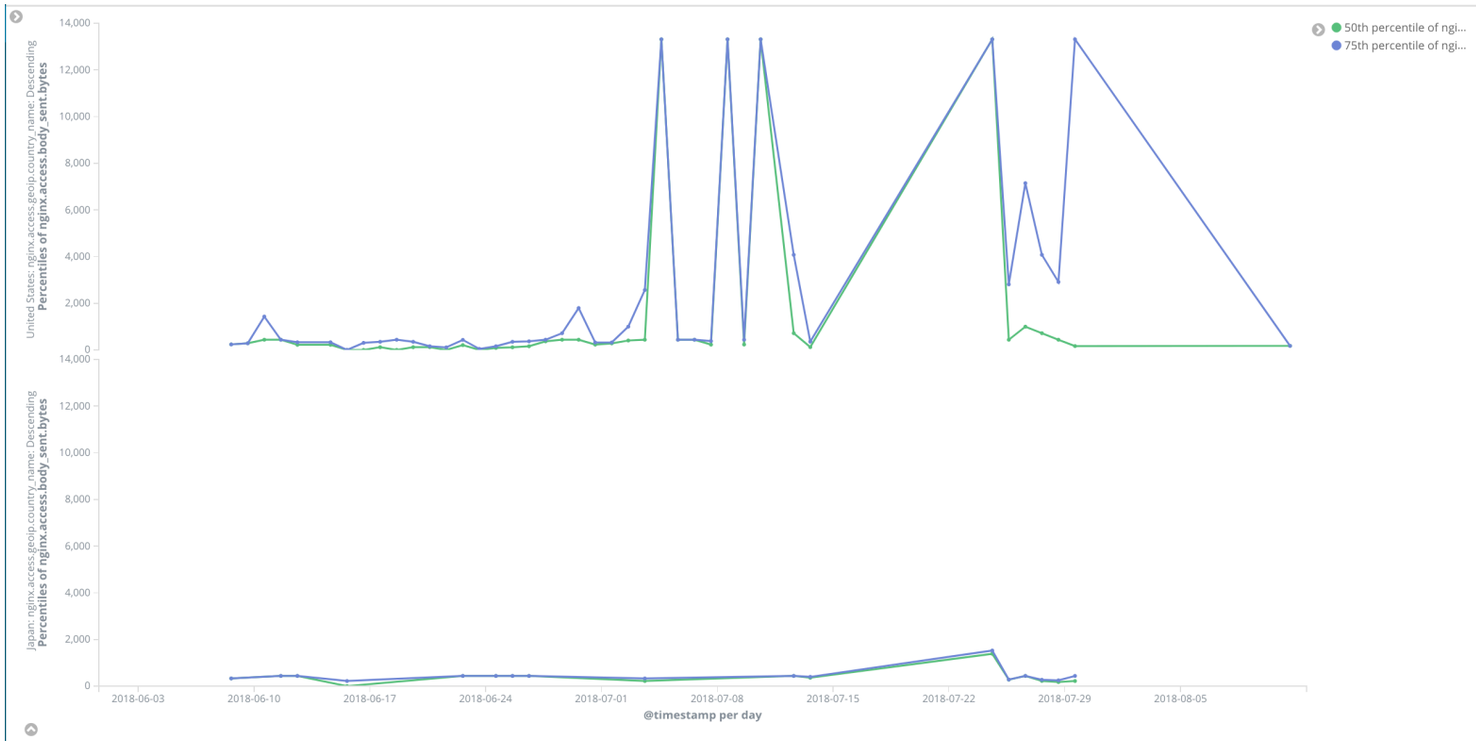
2SD Rule (2표준편차 법칙) 👑

심화



- 정규분포에서 약 95%의 값들이 평균에서 양쪽으로 2 표준편차 범위(μ±2σ)에 존재한다
- Kibana에서는 μ+2σ 값과 μ-2σ 값을 보여준 것이다 (변경 가능)

예제 10) Line Chart



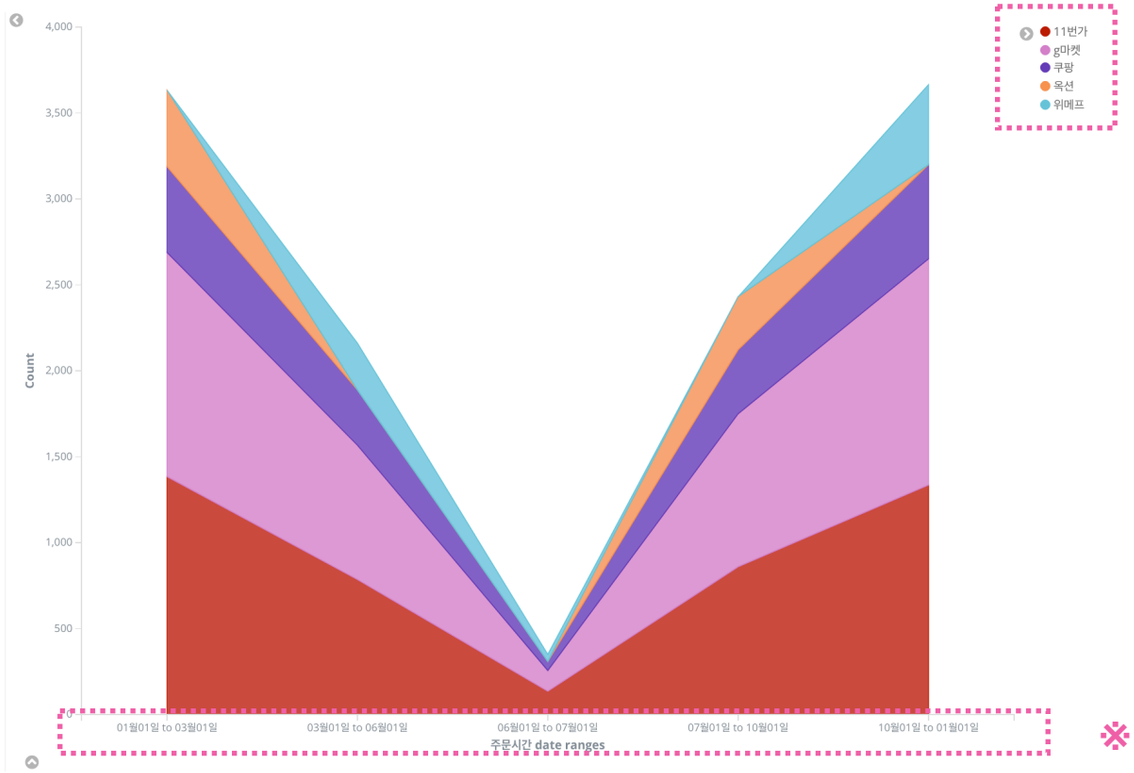
조건

- **nginx-*** index 중에서
- “@timestamp” field 기준 **“2018-06-11 ~ 2018-08-12”** documents를
- “**nginx.access.body_sent.bytes**” field의 **평균이 가장 큰** ┐ **(split chart) 국가별**
- “**nginx.access.geoip.country_name**” field **2개**에 대해서 ┘
- “**@timestamp**” field를 기준으로 **일별**(daily) bucket을 생성한 후 —— **(x축) 일별**
- “**nginx.access.body_sent.bytes**”의 **50백분위수**와 **75백분위수** —— **데이터 크기**

Area Chart

- x축의 변화에 따른 특정 field의 값의 추이를 볼 때 사용
- 주로 x축은 날짜/시간 관련 field를 사용
- area 내부도 이용할 수 있어 (line chart 대비) 한 차원 높은 분석 가능
- kibana 설정은 line // bar // area 모두 유사

예제 11) Area Chart



※ 5개가 표시되어야 한다

※ 날짜 format 무시

조건

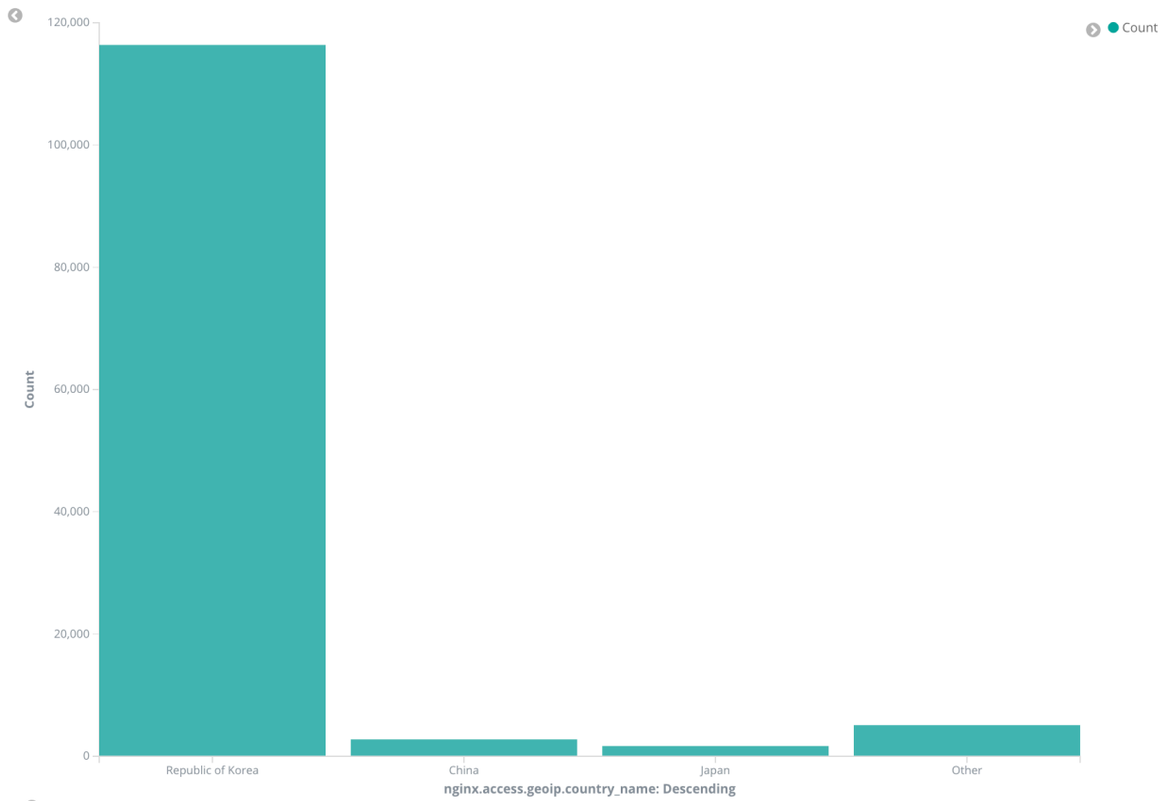
- **shopping** index 중에서
- "주문시간" field 기준 **this year** documents를
- "**주문시간**" field 값을 기준으로 아래와 같은 bucket을 생성한 후
  - 2018-01-01 ~ 2018-03-01
  - 2018-03-01 ~ 2018-06-01
  - 2018-06-01 ~ 2018-07-01
  - 2018-07-01 ~ 2018-10-01
  - 2018-10-01 ~ 2019-01-01

  → **(x축) 특정 기간 별**
- "**상품가격**" field의 **합이 가장 큰 4개**의 "**구매사이트**" field 별 → **(Split series) 구매사이트 별**
- documents **개수** → **판매수**

Vertical Bar

- x축의 변수에 따른 특정 field 값을 시각화 할 때 사용
- 필수는 아니나 x축에는 주로 ordinal value 사용 (예: 20~29, 30~39, 40~49 등)
- kibana 설정은 line // bar // area 모두 유사

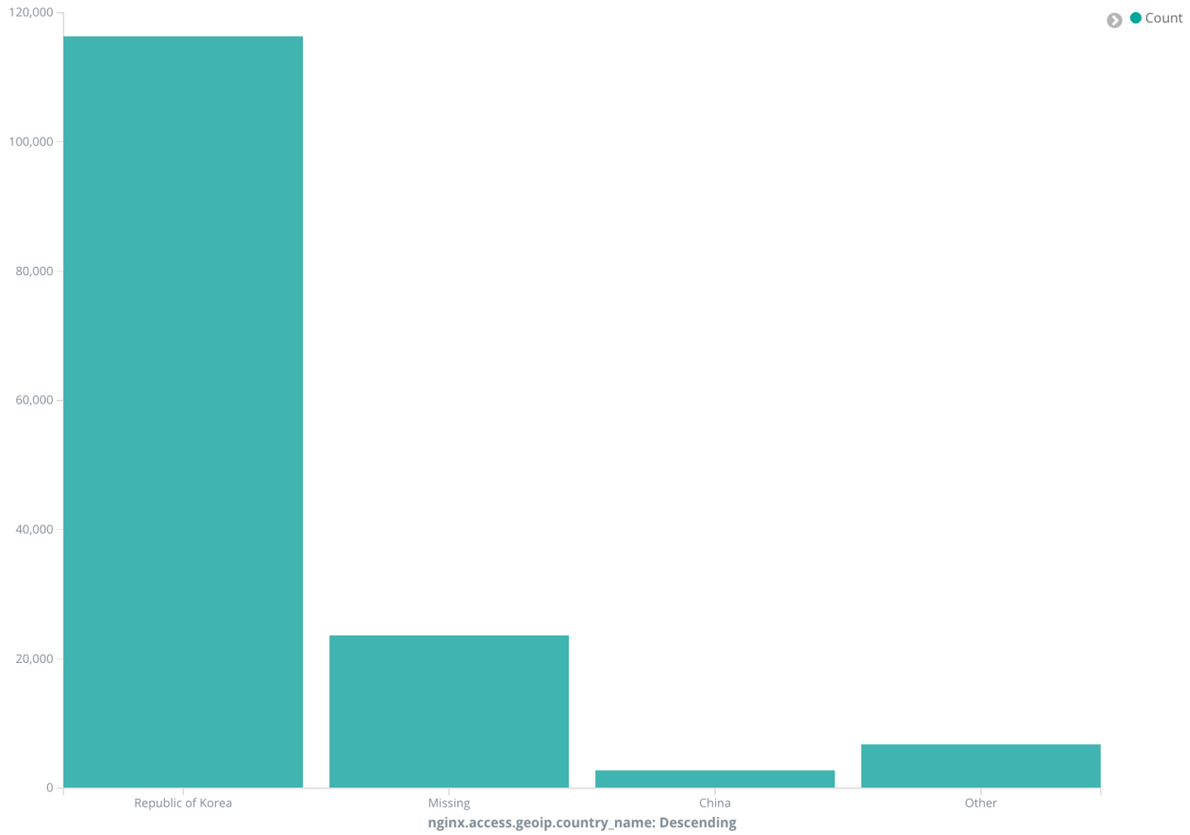
예제 12) Vertical Bar



조건

- **nginx-*** index 중에서
- “주문시간” field 기준 **”2018-06-01 ~ 2018-08-12”** documents를
- **documents 개수가 가장 많은 3개**의 **nginx.access.geoip.country_name** 별 —— **국가별**
- documents **개수** ┐ **접속수**
- (단, 그 외의 nginx.access.geoip.country_name는 **others**로 묶어서 표시) ┘

예제 13) Vertical Bar



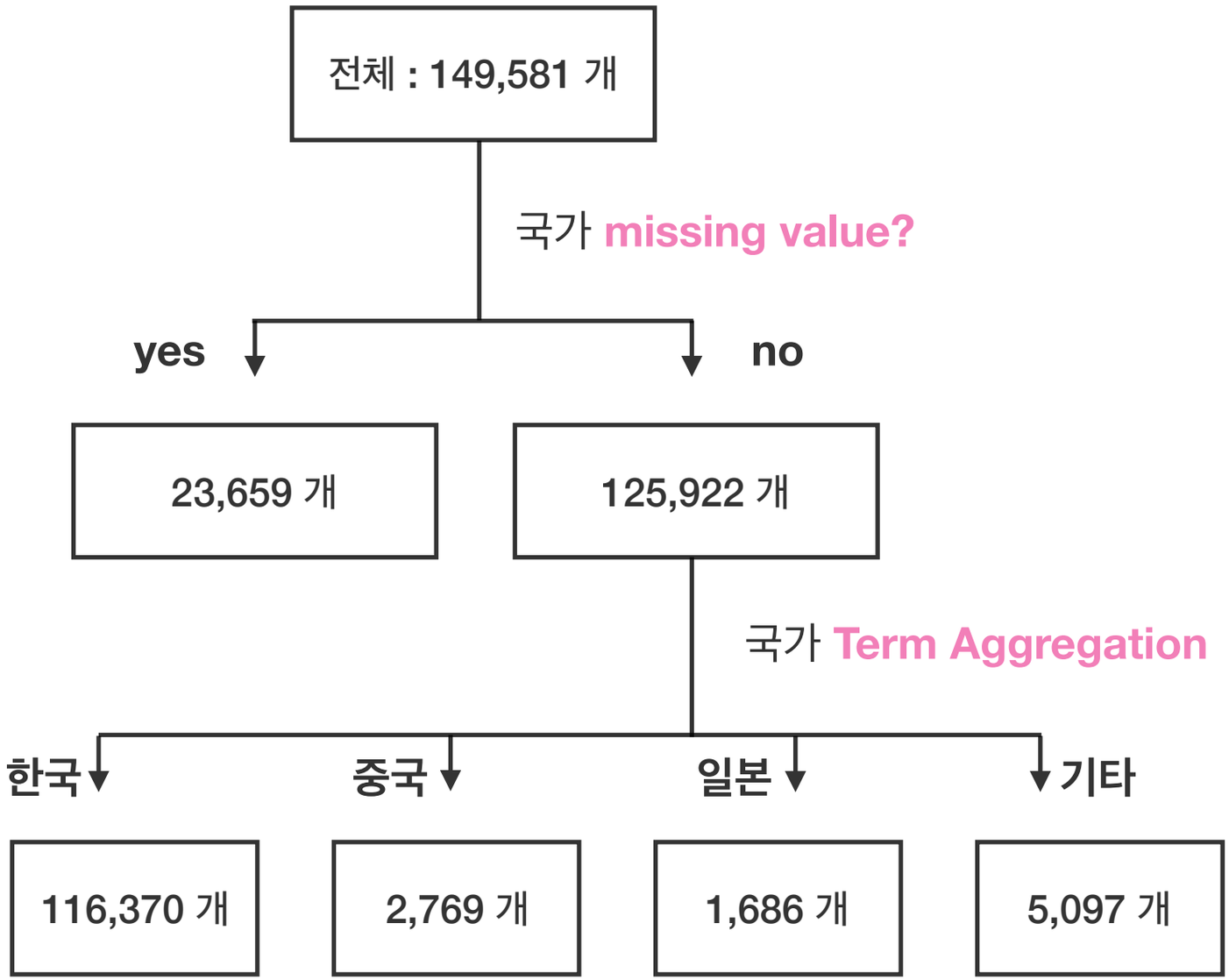
조건

- **nginx-*** index 중에서
- “주문시간” field 기준 **”2018-06-01 ~ 2018-08-12”** documents를
- **documents 개수가 가장 많은 3개**의 **nginx.access.geoip.country_name** 별 — **국가별**
- documents **개수**
- 단
  - 그 외 **nginx.access.geoip.country_name** field는 **others**로 묶어서 표시
  - 만약 **nginx.access.geoip.country_name** field가 **missing**일 경우 별도 표시

접속수

잠깐3, 옵션을 제대로 보고 넘어가자

데이터 현황


전체 : 149,581 개
국가 missing value?
yes
no
23,659 개
125,922 개
국가 Term Aggregation
한국
중국
일본
기타
116,370 개
2,769 개
1,686 개
5,097 개

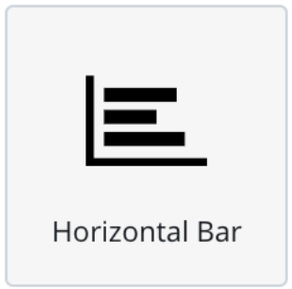
심화

미세하게 다르게 나오면 👑

missing option	others option	(표시되는) missing 개수	(표시되는) others 개수
☐ Show missing values	☐ Group other values in separate bucket	0	0
☐ Show missing values	☑ Group other values in separate bucket	0	5,097
☑ Show missing values	☐ Group other values in separate bucket	23,659	0
☑ Show missing values	☑ Group other values in separate bucket	23,659	6,783

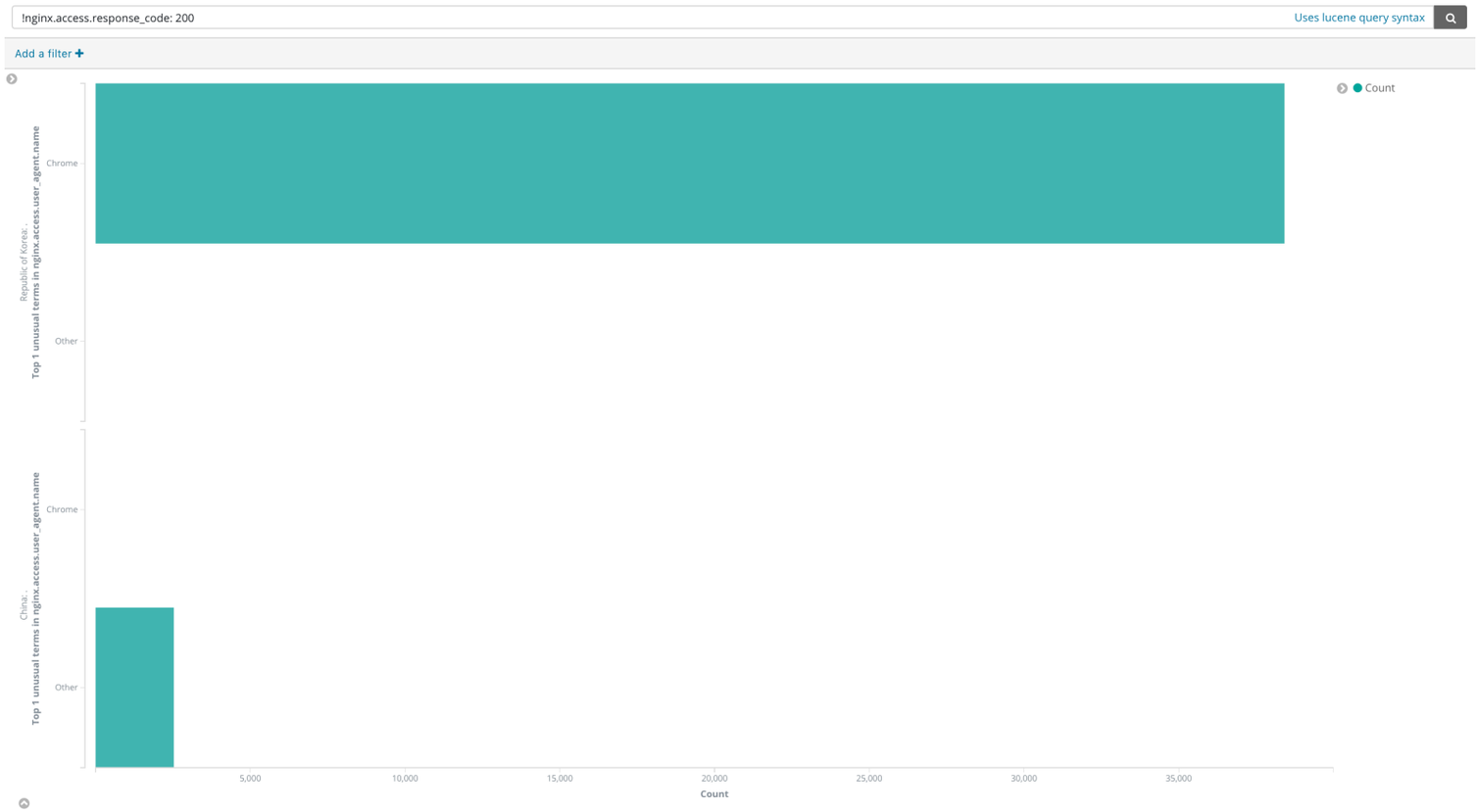
한국,중국, missing 제외

Horizontal Bar



- Vertical Bar와 유사한 용도
- 단, Vertical Bar와는 달리 x축에는 주로 nominal value를 사용 (예: 한국/중국/일본)
- kibana 설정은 line // bar // area 모두 유사

예제 14) Horizontal Bar

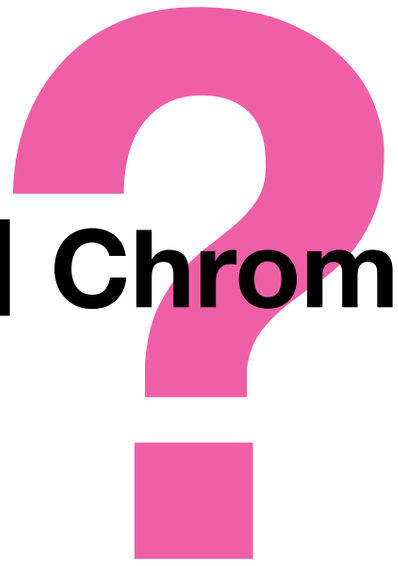


조건

- **nginx-*** index 중에서
- “@timestamp” field 기준 **”2018-06-01 ~ 2018-08-12”** documents로 필터링한 후의
- **documents 개수가 가장 많은 2개**의 **nginx.access.geoip.country_name** 별 — **(Split Chart) 국가별**
- **전체 documents 대비**
- **“nginx.access.response_code” field가 200이 아닌** documents 중에서
- **특별한 “nginx.access.user_agent.name”** field로 bucket을 **1개** 생성한 후

  (x축)
  (응답코드가 200이 아닌 데이터 중에서)
  특별한 user_agent.name 별
- 각 bucket별 document **개수** — **접속수**

잠깐4, 한국과 중국에서 Chrome과 Others는 왜 나왔지?



한국

user_agent.name	background		foreground		증감
Chrome	115,129	98.94%	38,417	98.21%	**0.73%**
IE	486	0.42%	320	0.82%	-0.40%
Mobile Safari UI/WKWebView	319	0.27%	191	0.49%	-0.21%
Other	101	0.09%	34	0.09%	-0.00%
Chrome Mobile	150	0.13%	91	0.23%	-0.10%
Chrome Mobile iOS	126	0.11%	46	0.12%	-0.01%
Firefox	2	0.00%	1	0.00%	-0.00%
curl	28	0.02%	12	0.03%	-0.01%
FacebookBot	17	0.01%	0	0.00%	0.01%
Mobile Safari	8	0.01%	6	0.02%	-0.01%
Total	116,366	100%	39,118	100%	

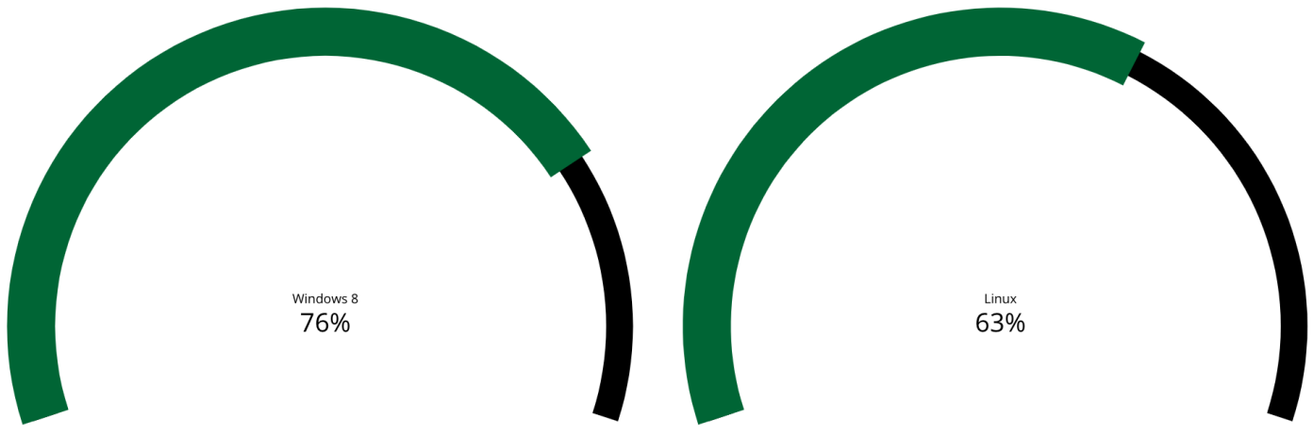
중국

user_agent.name	background		foreground		증감
Other	2,646	95.56%	2,528	97.95%	**-2.39%**
Chrome	43	1.55%	16	0.62%	0.93%
Sogou Explorer	32	1.16%	3	0.12%	1.04%
IE	29	1.05%	19	0.74%	0.31%
Firefox	17	0.61%	14	0.54%	0.07%
Python Requests	1	0.04%	1	0.04%	-0.00%
masscan	1	0.04%	0	0.00%	0.04%
Total	2,769	100%	2,581	100%	

Goal

- 직접 설정한 목표 대비 현재 어느 정도 달성했는지 시각화 할 때 사용

Goal Object



Average nginx.access.body_sent.bytes

조건

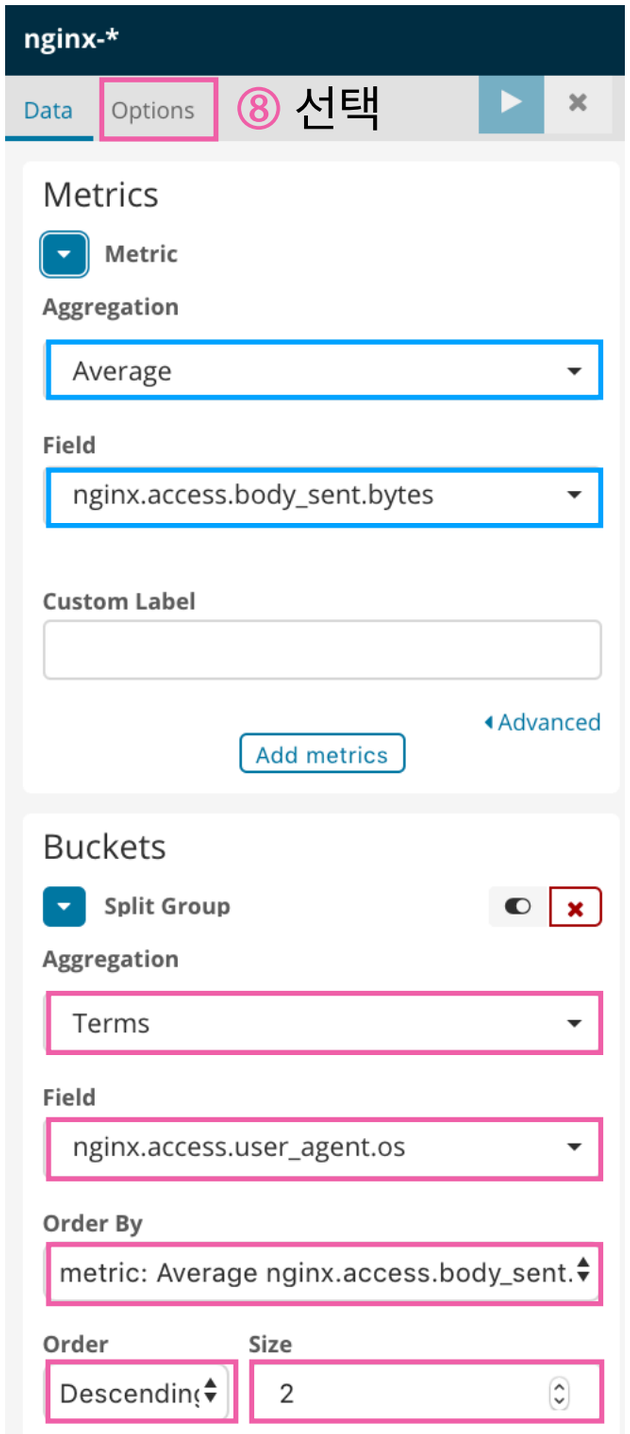
- **nginx-*** index 중에서
- “@timestamp” field 기준 **”2018-06-01 ~ 2018-08-12”** documents의
- “**nginx.access.body_sent.bytes**”의 **평균이 가장 높은**
- **“nginx.access.user_agent.os”** field **2개** 별
- “**nginx.access.body_sent.bytes**” field의 **평균**의
- **목표를 100,000** bytes 으로 설정했을 때의 **달성율**

os 별

요청 데이터 평균값의

목표 달성율

Goal Configuration 1



① Average Aggregation 선택

② Average Aggregation 적용할 Field 선택

③ Terms Aggregation 선택

④ Terms Aggregation 적용할 Field 선택

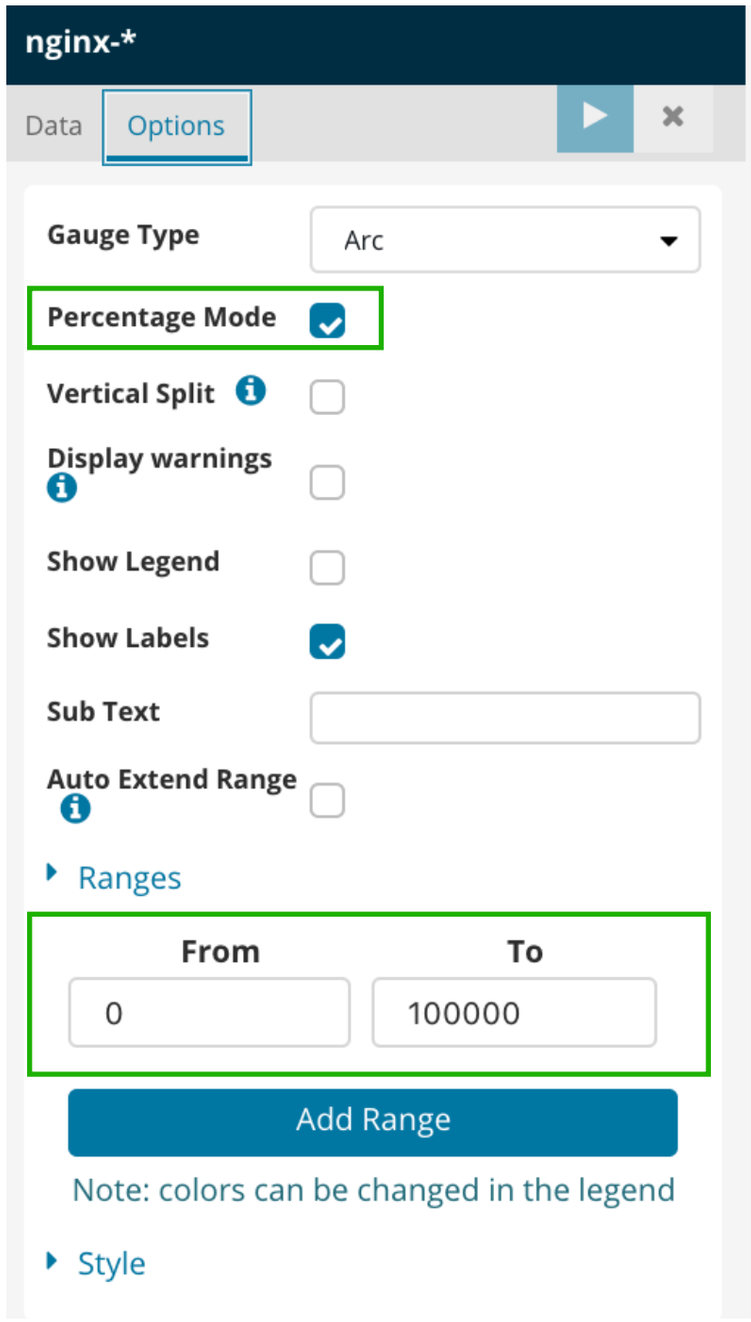
⑤ bucket 선정을 위한 metric 선택

⑥ 반영할 bucket 개수 입력

⑦ 정렬방식 선택 (오름/내림)

Goal Configuration 2

⑨ 백분율로 표시하기 위해 선택

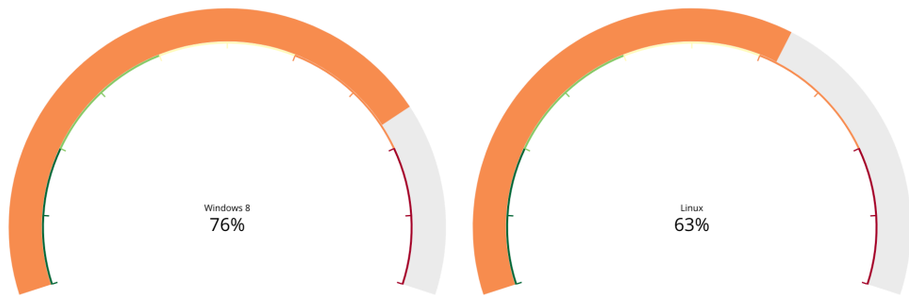


⑩ 목표치 설정

Gauge

- 직접 설정한 목표 대비 현재 어느 정도 달성했는지 단계별로 시각화 할 때 사용

Gauge Object



조건

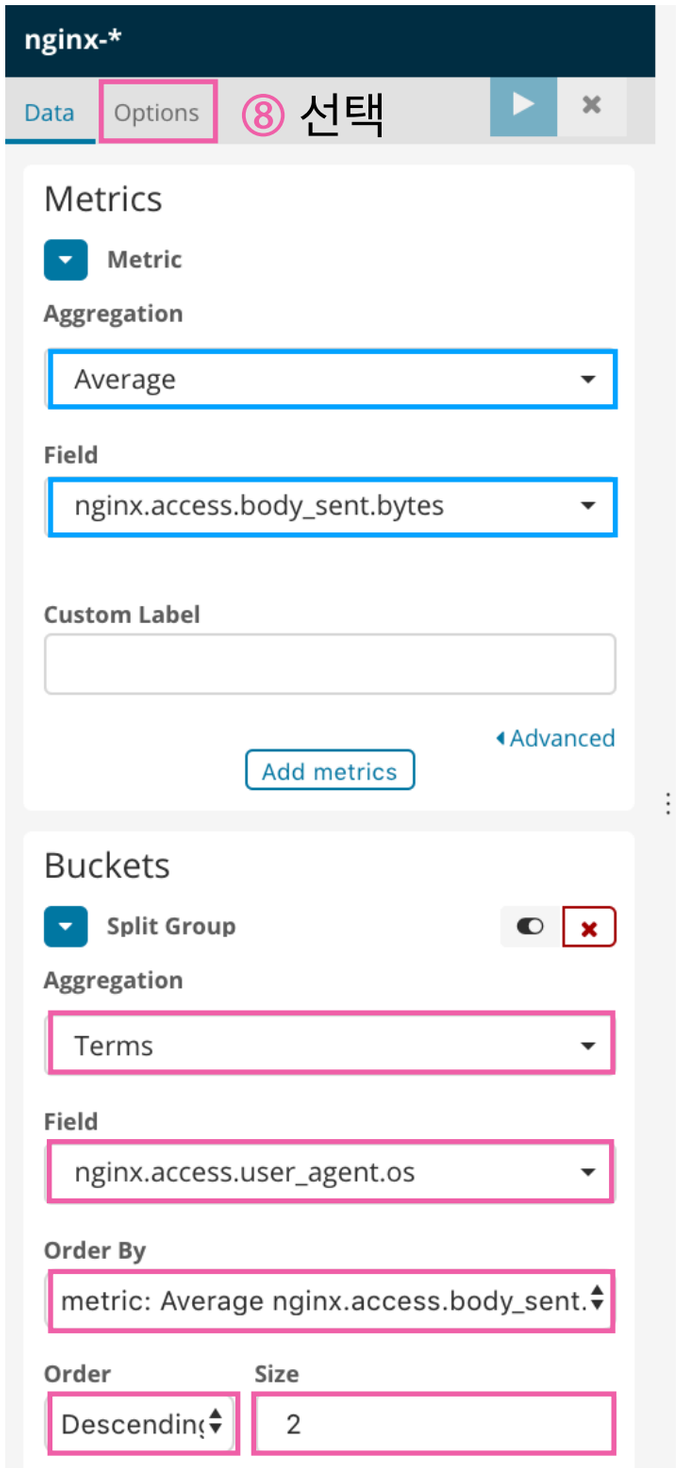
- **nginx-*** index 중에서
- “@timestamp” field 기준 **”2018-06-01 ~ 2018-08-12”** documents의
- “**nginx.access.body_sent.bytes**”의 **평균이 가장 높은**
- **“nginx.access.user_agent.os”** field **2개** 별
- “**nginx.access.body_sent.bytes**” field의 **평균**의
- **단계별 목표**를 아래와 같이 설정했을 때의 **달성율**
  - **0 ~ 20000**
  - **20000 ~ 40000**
  - **40000 ~ 60000**
  - **60000 ~ 80000**
  - **80000 ~ 100000**

os 별

요청 데이터 평균값의

(단계별) 목표 달성율

Gauge Configuration 1



① Average Aggregation 선택

② Average Aggregation 적용할 Field 선택

③ Terms Aggregation 선택

④ Terms Aggregation 적용할 Field 선택

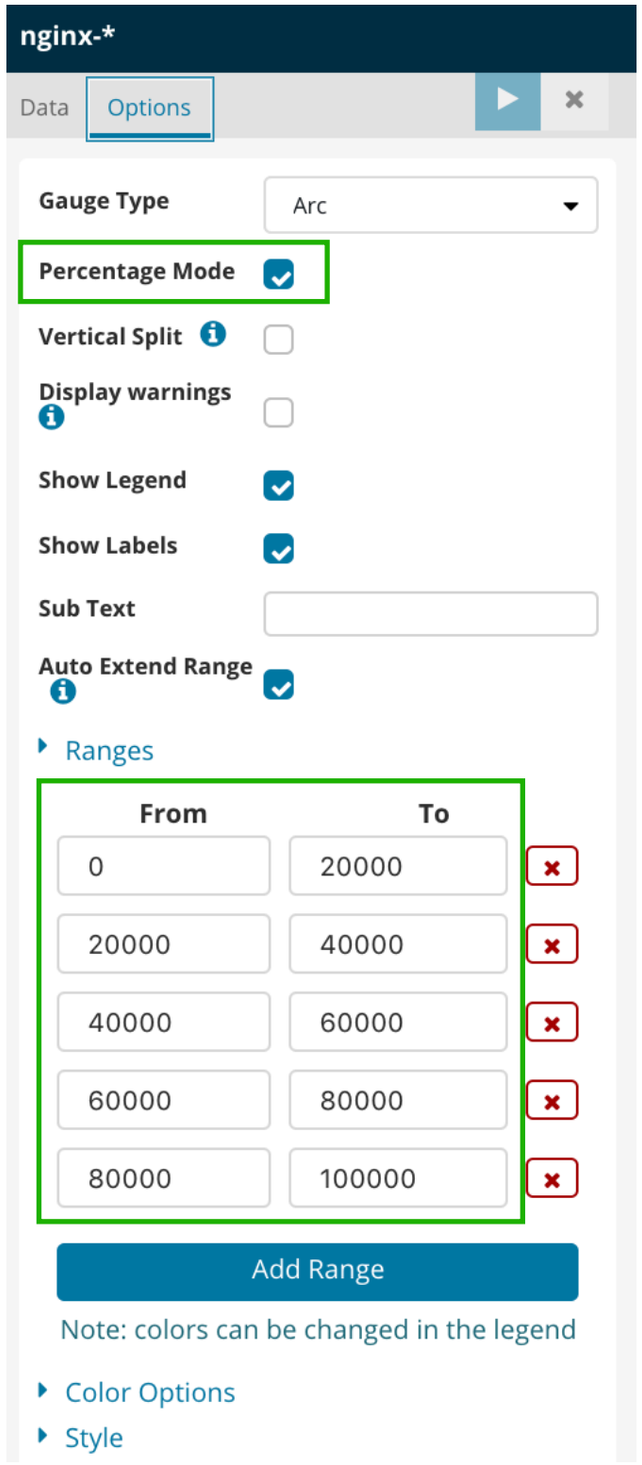
⑤ bucket 선정을 위한 metric 선택

⑥ 반영할 bucket 개수 입력

⑦ 정렬방식 선택 (오름/내림)

Gauge Configuration 2

⑨ 백분율로 표시하기 위해 선택



⑩ 목표치 설정

Data Table

- Excel과 같은 테이블 형태로 시각화
- 데이터가 분산되어 있어 차트와 같은 형태로는 보기 어려운 경우 유용

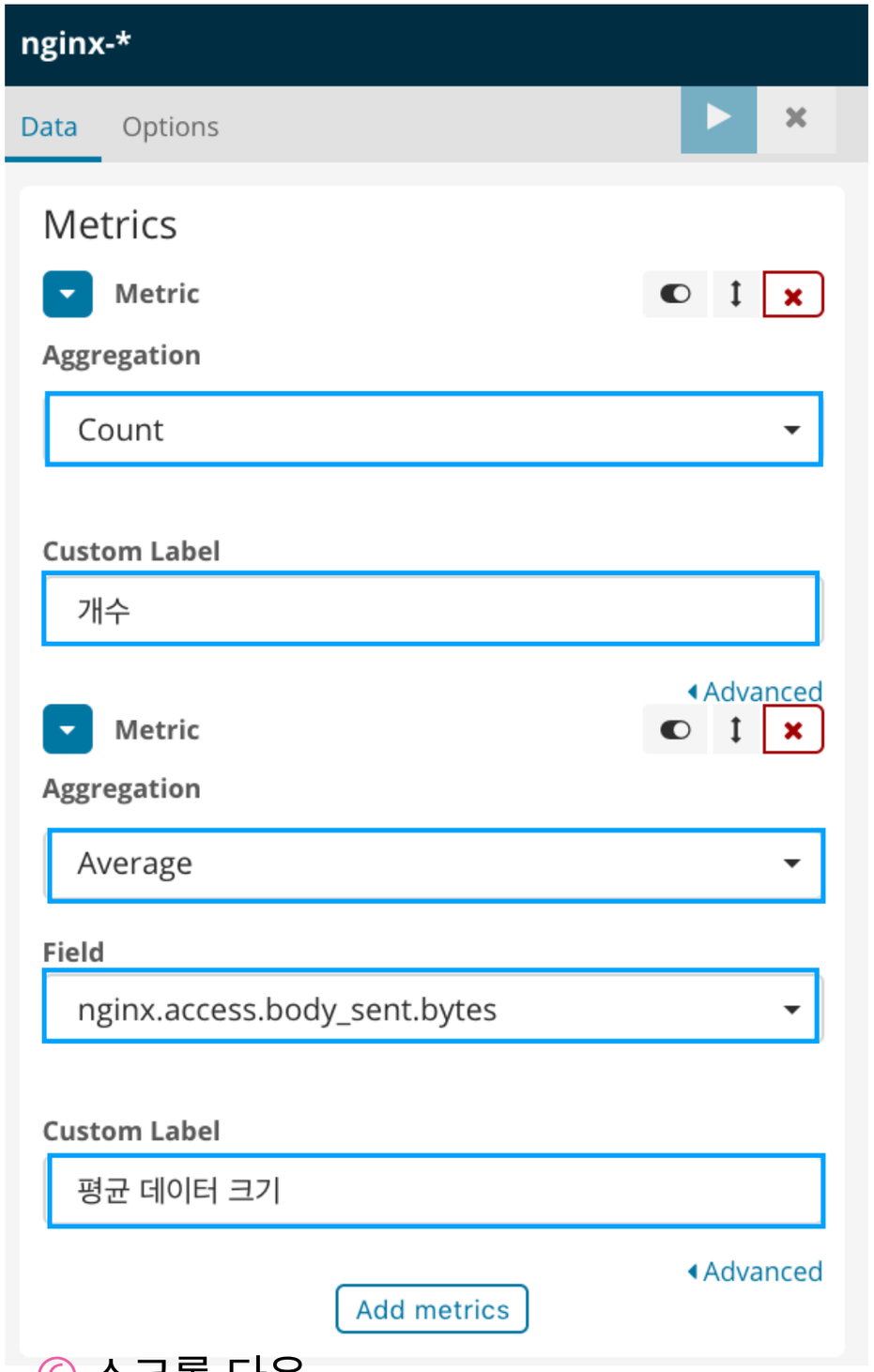
Data Table Object

날짜	도시	데이터 크기	os	개수	평균 데이터 크기
2018-06-09	Seoul	0 to 1,000	iOS	34	152.529
2018-06-09	Seoul	1,000 to +∞	iOS	20	145,340.5
2018-06-09	Quito	0 to 1,000	Windows 10	2	328
2018-06-10	Seoul	0 to 1,000	Mac OS X	1,553	406.161
2018-06-10	Seoul	0 to 1,000	iOS	100	39.75
2018-06-10	Seoul	1,000 to +∞	Mac OS X	140	3,624.15
2018-06-10	Seoul	1,000 to +∞	iOS	18	169,053.444
2018-06-10	Incheon	0 to 1,000	iOS	75	121.107
2018-06-10	Incheon	1,000 to +∞	iOS	8	9,340.875
2018-06-11	Seoul	0 to 1,000	Mac OS X	30	415.267

조건

- **nginx-*** index 중에서
- “@timestamp” field 기준 **”2018-06-01 ~ 2018-08-12”** documents의
- “**@timestamp**” field로 일별(daily) bucket을 만들고 — **날짜별**
- **documents 개수가 많은** “**nginx.access.geoip.city_name**” field **2개** 별 — **도시별**
- “**nginx.access.body_sent.bytes**” field로 아래와 같이 bucket을 생성한 후 — **데이터 크기별**
  - **0 ~ 1000**
  - **1000 ~**
- **document 개수가 많은** “**nginx.access.user_agent.os**” field **2개** 별 — **os 별**
- **document 개수**와 — **접속수**
- “**nginx.access.body_sent.bytes**” field 값의 **평균** — **평균 데이터 크기**

Data Table Configuration 1



① Count Aggregation 선택

②
Data Table Column 이름에
표시될 Label 입력

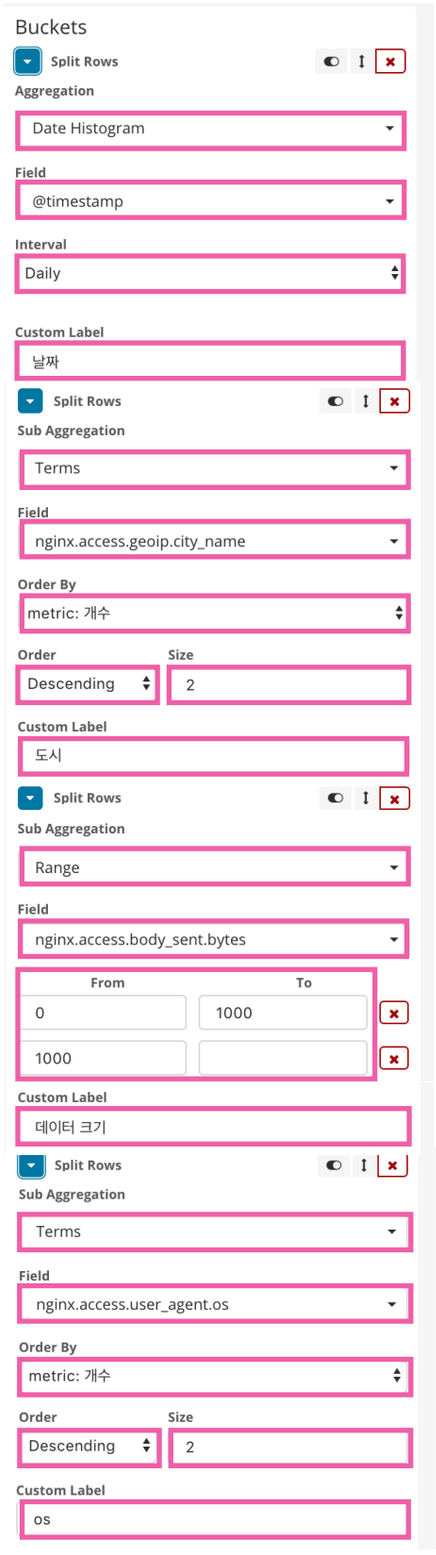
③ Average Aggregation 선택

④
Average Aggregation
적용할 Field 선택

⑤
Data Table Column 이름에
표시될 Label 입력

⑥ 스크롤 다운

Data Table Configuration 2



⑦ Date Histogram Aggregation 선택

⑧ Date Histogram Aggregation 적용할 Field 선택

⑨ Date Histogram 간격 선택

⑩ Data Table Column 이름에 표시될 이름 입력

⑪ Terms Aggregation 선택

⑫ Terms Aggregation 적용할 Field 선택

⑬ Terms Aggregation 후 bucket을 정렬할 때 사용할 metric 선택

⑭ 사용할 bucket 개수 입력

⑮ 정렬방식 선택 (오름/내림)

⑯ Data Table Column 이름에 표시될 이름 입력

⑰ Range Aggregation 선택

⑱ Range Aggregation 적용할 Field 선택

⑲ Range Aggregation bucket 구간 직접 입력

⑳ Data Table Column 이름에 표시될 이름 입력

㉑ Terms Aggregation 선택

㉒ Terms Aggregation 적용할 Field 선택

㉓ Terms Aggregation 후 bucket을 정렬할 때 사용할 metric 선택

㉔ 사용할 bucket 개수 입력

㉕ 정렬방식 선택 (오름/내림)

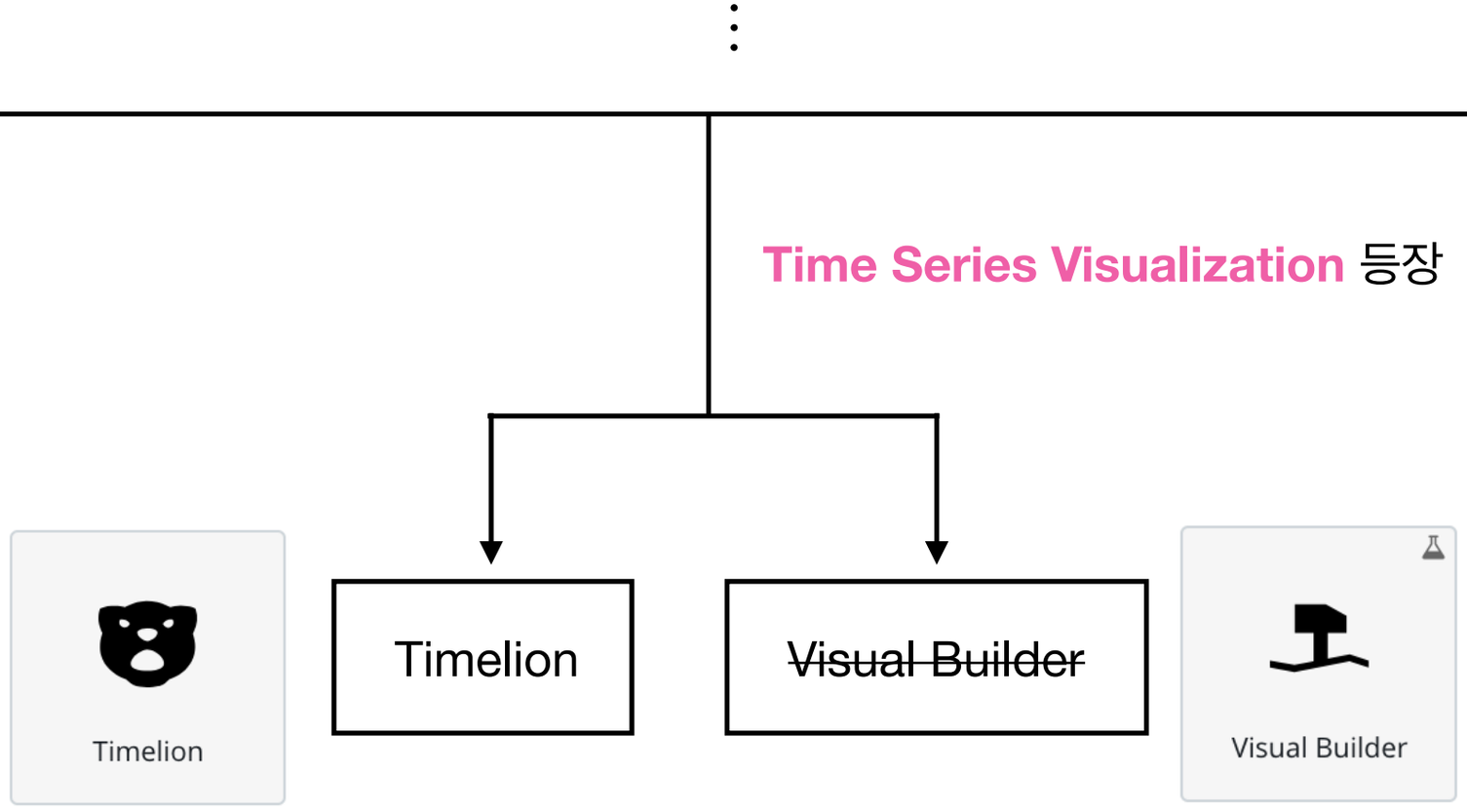
㉖ Data Table Column 이름에 표시될 이름 입력

Line/Bar/Area Chart로 시계열 데이터를 시각화하면서 아쉬운 점?

- 서로 다른 Index (Pattern) 의 데이터를 하나의 Visualization으로 시각화 못한다
- 특정 Field Value의 기준시점과 비교시점 값을 시각적으로 비교하기 어렵다

⋮

Time Series Visualization 등장



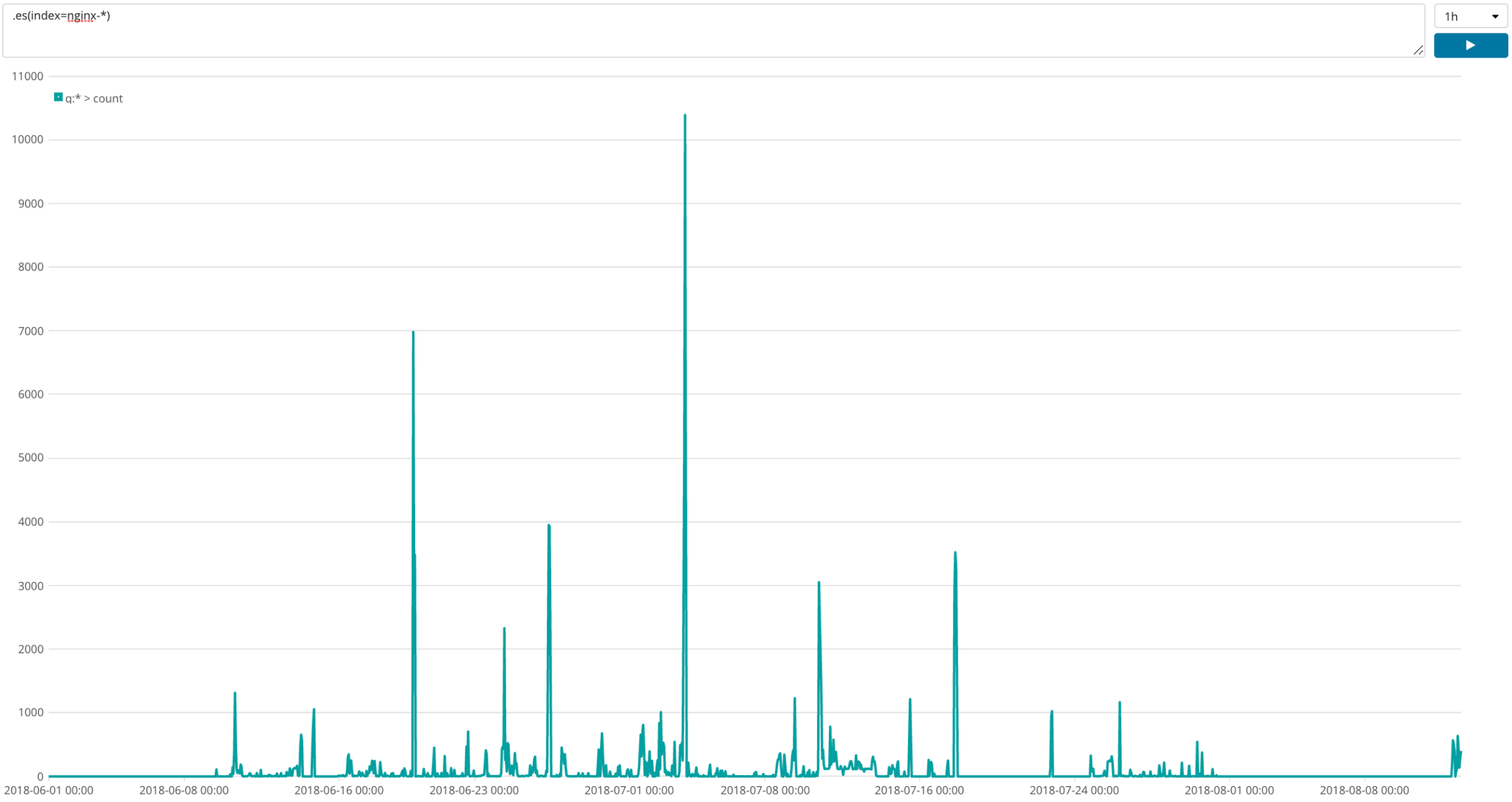
기본적으로 아래와 같은 함수를 제공한다 👑

목록

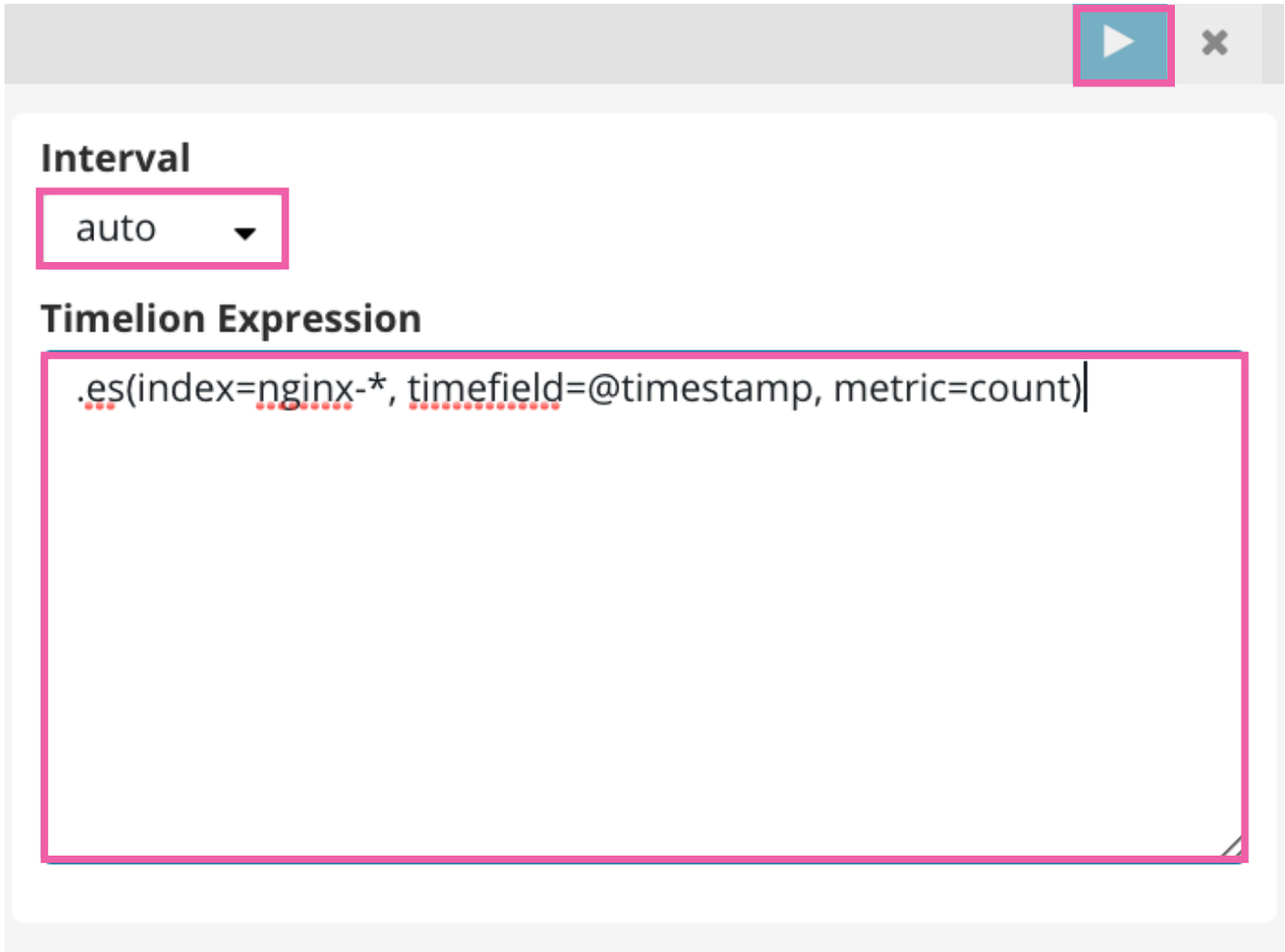
- 기본함수 - `.es()`
- 조건함수 - `.es().if()`
- 수학함수 - `.es().multiply()`
- 수학함수 - `.es().divide()`
- 수학함수 - `.es().subtract()`
- 수학함수 - `.es().sum()`
- 수학함수 - `.es().abs()`
- 수학함수 - `.es().log()`
- 수학함수 - `.es().max()`
- 수학함수 - `.es().min()`
- 수학함수 - `.es().static()`
- 수학함수 - `.es().cusum()`
- 수학함수 - `.es().derivative()`
- 수학함수 - `.es().movingaverage()`
- 수학함수 - `.es().scale_interval()`
- 수학함수 - `.es().range()`
- 수학함수 - `.es().trend()`
- 스타일함수 - `.es().bars()`
- 스타일함수 - `.es().lines()`
- 스타일함수 - `.es().points()`
- 스타일함수 - `.es().label()`
- 스타일함수 - `.es().color()`
- 스타일함수 - `.es().yaxis()`
- 스타일함수 - `.es().title()`

시계열 데이터를 시각화 하는 데 있어 **기존 Visualization 과의 차이**에 집중하자

간단한 사용법을 보자


.es(index=nginx-*)
1h
11000
q:* > count
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
2018-06-01 00:00
2018-06-08 00:00
2018-06-16 00:00
2018-06-23 00:00
2018-07-01 00:00
2018-07-08 00:00
2018-07-16 00:00
2018-07-24 00:00
2018-08-01 00:00
2018-08-08 00:00

간단한 사용법을 보자


③ 실행
Interval
① Interval 선택
auto
Timelion Expression
② Timelion Expression 작성
.es(index=nginx-*, timefield=@timestamp, metric=count)

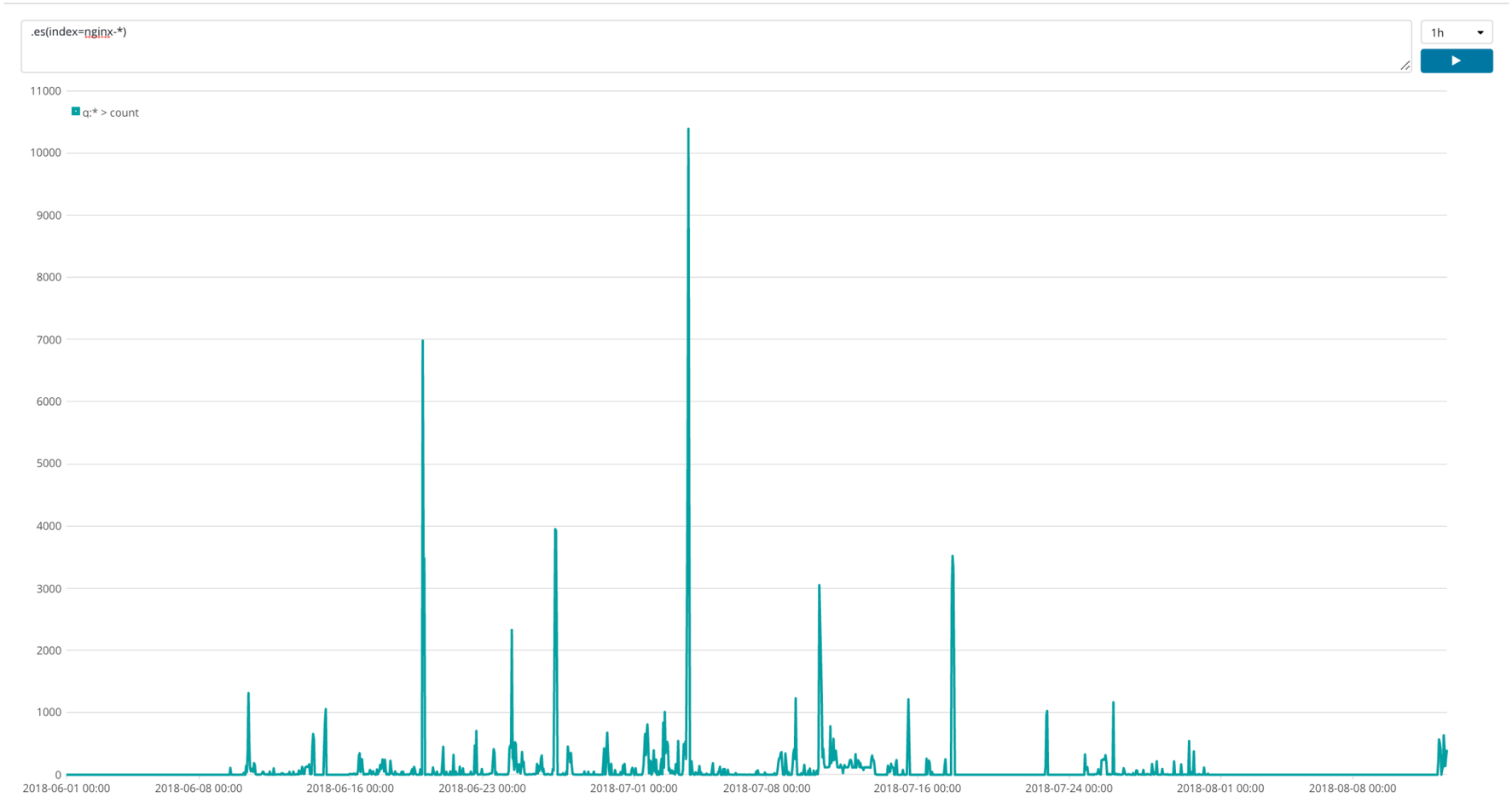
아래와 같이 설정을 고정하자

- Time Range : 2018-06-11 ~ 2018-08-12
- Interval : 1h
- 스크립트 : 👑

Timelion - Index 설정 👑

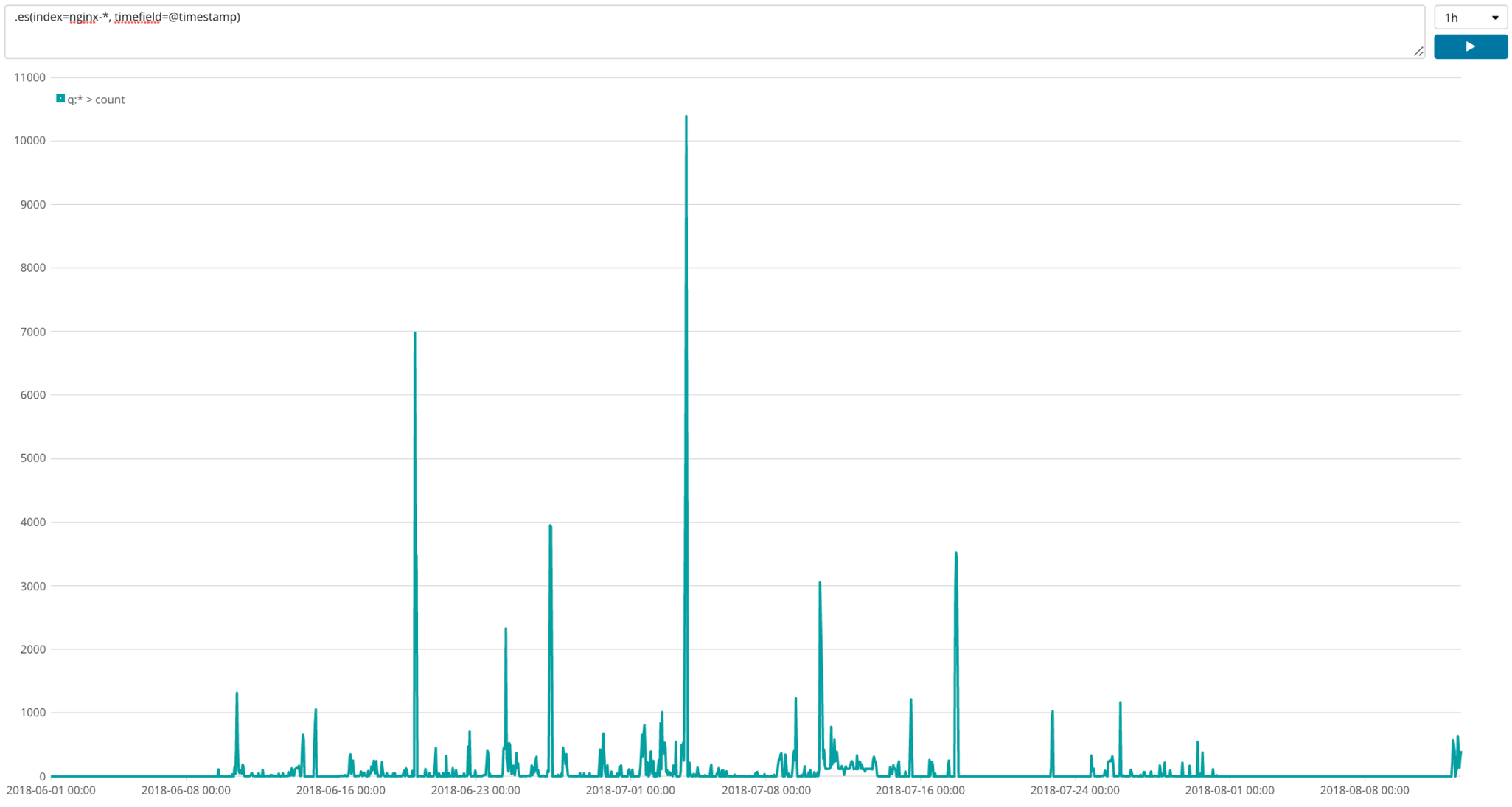
.es(index=nginx-*)

※ Kibana에서 Index Patterns에 등록하지 않고도 사용할 수 있다



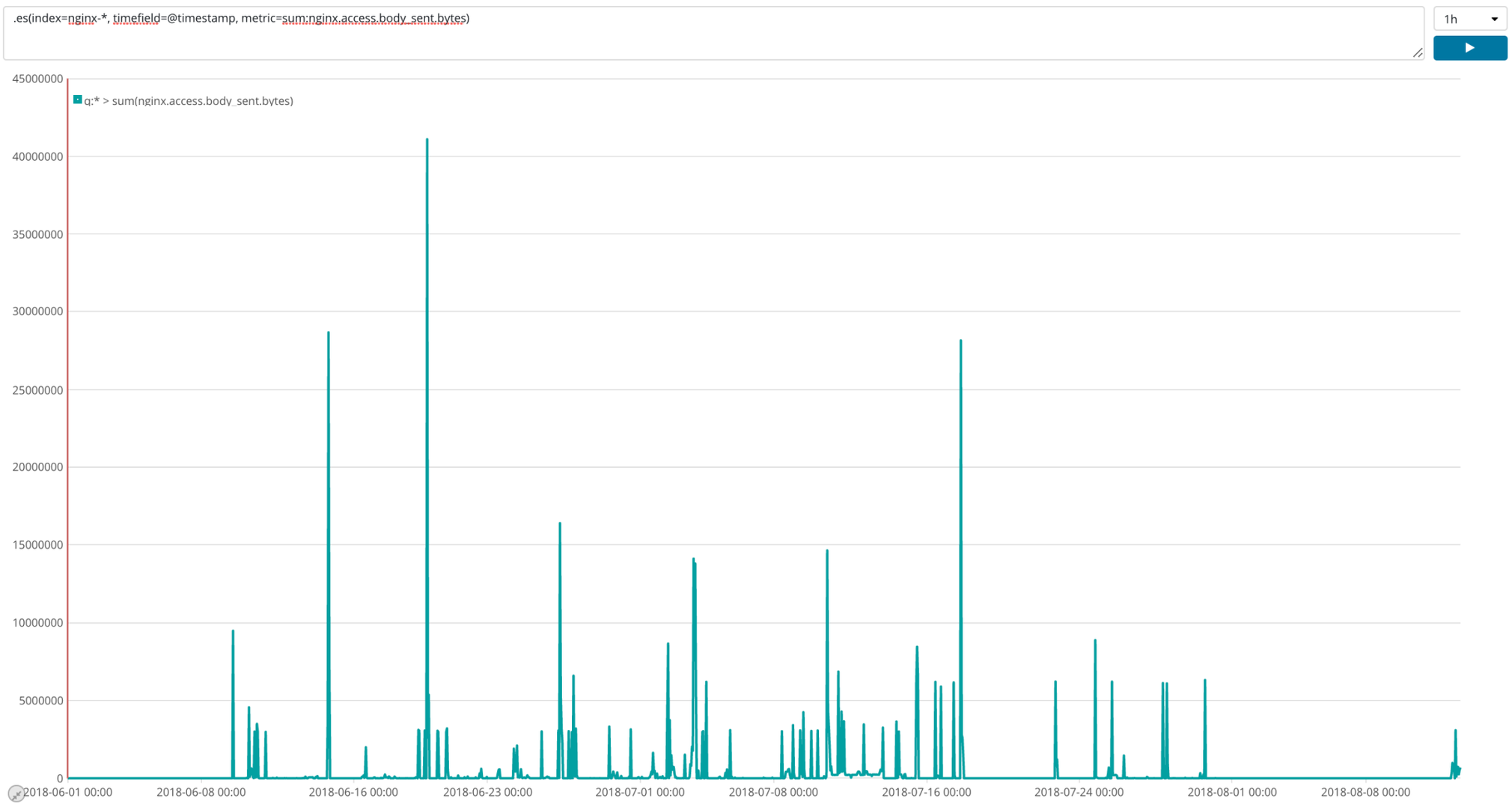
Timelion - timefield 설정 👑

.es(index=nginx-*, timefield=@timestamp) **※ default : @timestamp**



Timelion - metric 설정 👑

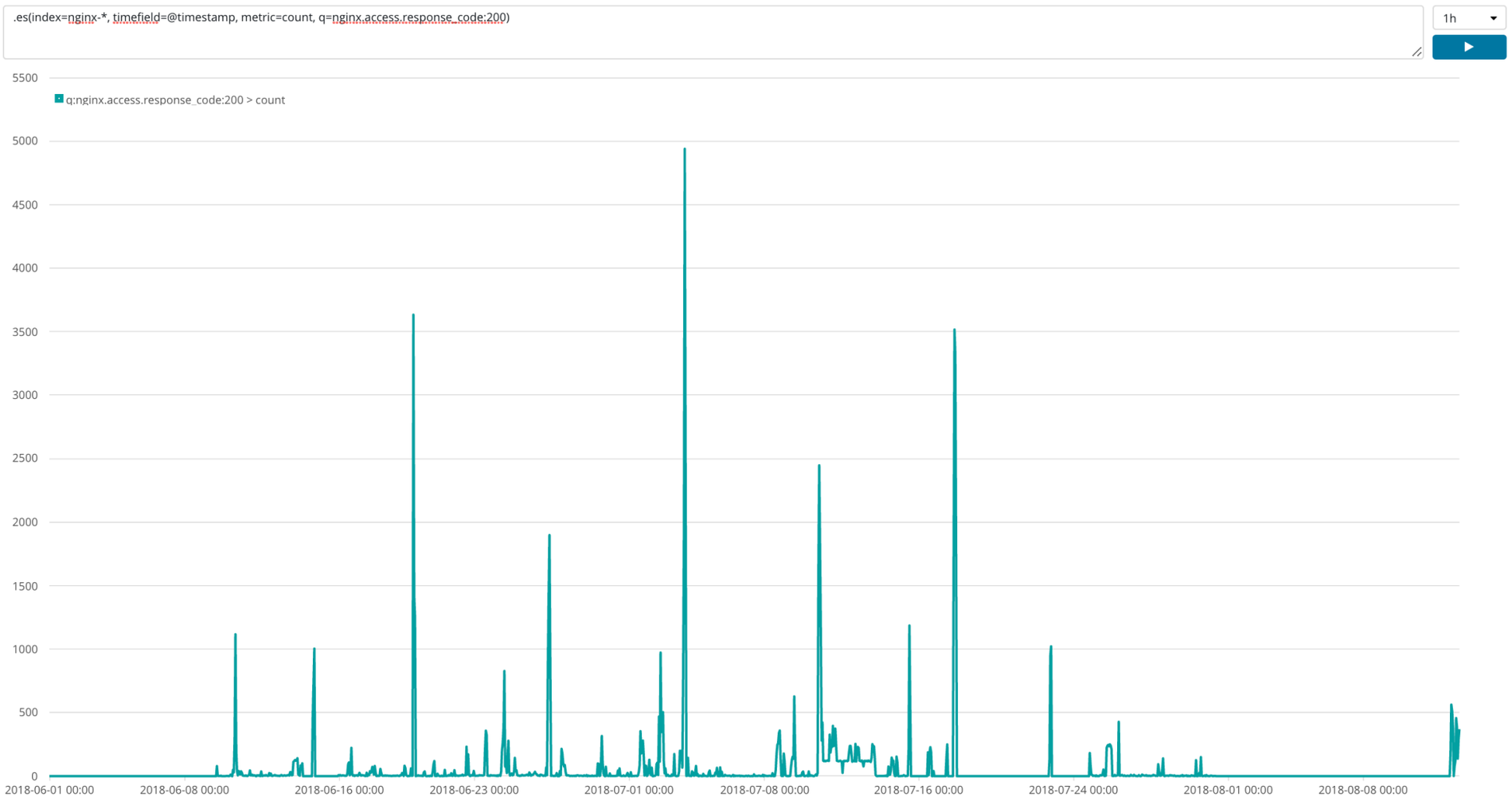
.es(index=nginx-*, timefield=@timestamp, metric=sum:nginx.access.body_sent.bytes) **※ default : count**



Timelion - query 설정 👑

.es(index=nginx-*, timefield=@timestamp, metric=count, q=nginx.access.response_code:200)

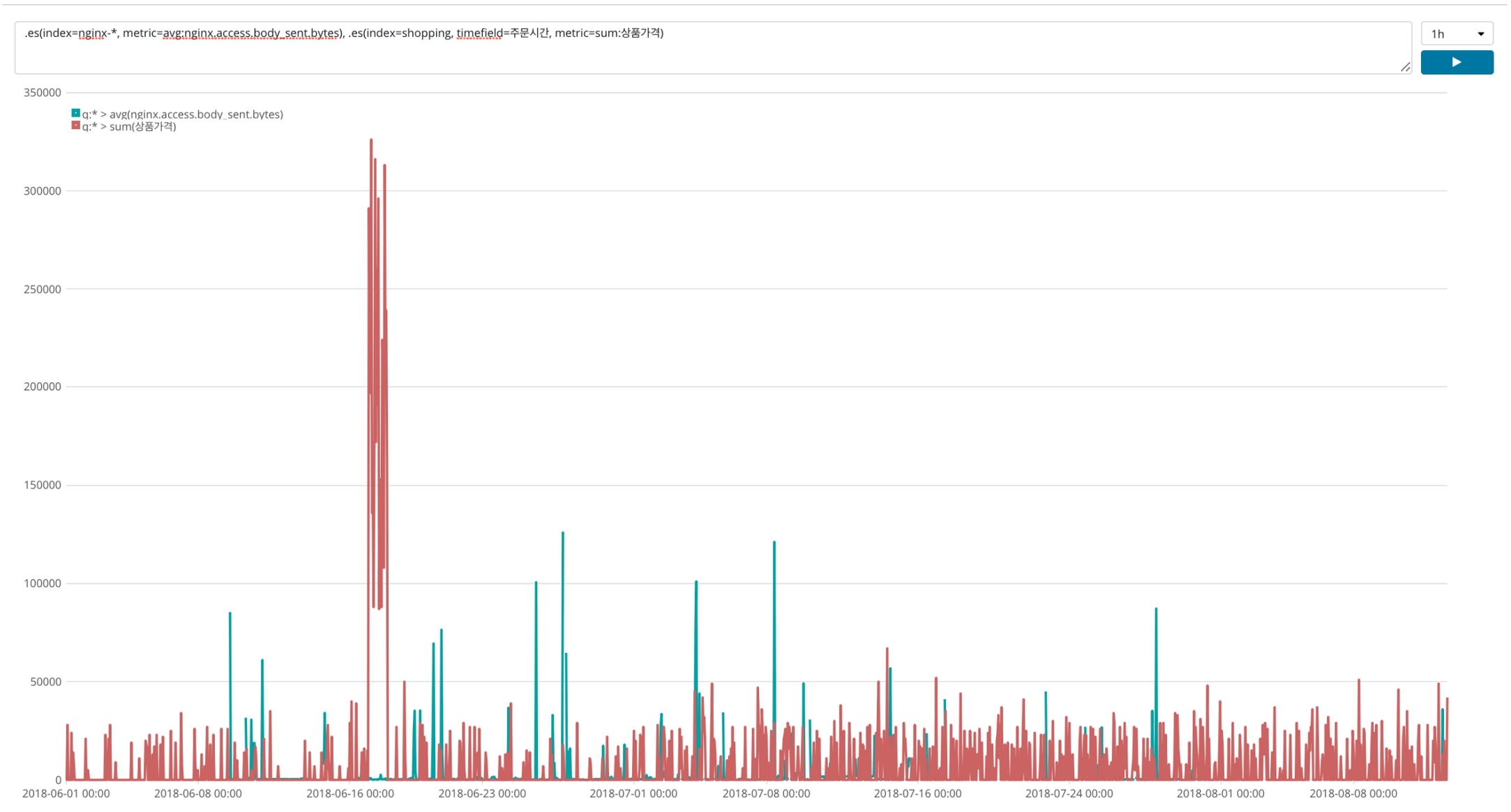
※ 문법 배울 예정



Timelion - multiple timelion 👑

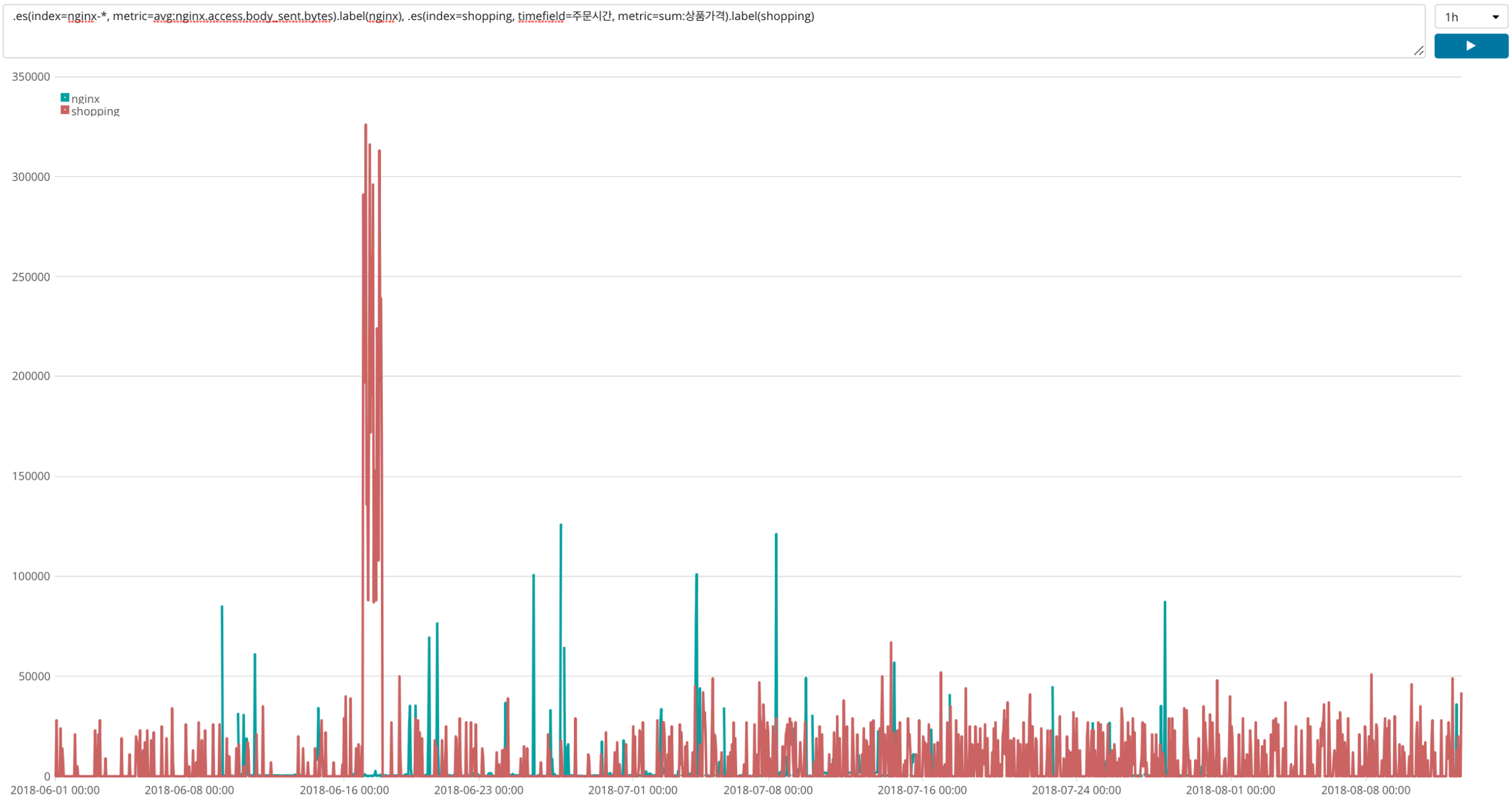
```
.es(index=nginx-*, metric=avg:nginx.access.body_sent.bytes) , .es(index=shopping, timefield=주문시간, metric=sum:상품가격)
```

※ nginx-*과 shopping과 같이 전혀 연관 없어 보이는 index도 통합해서 시각화



Timelion - label 👑

```
.es(index=nginx-*, metric=avg:nginx.access.body_sent.bytes).label(nginx) , .es(index=shopping, timefield=주문시간, metric=sum:상품가격).label(shopping)
```



Timelion - divide 👑

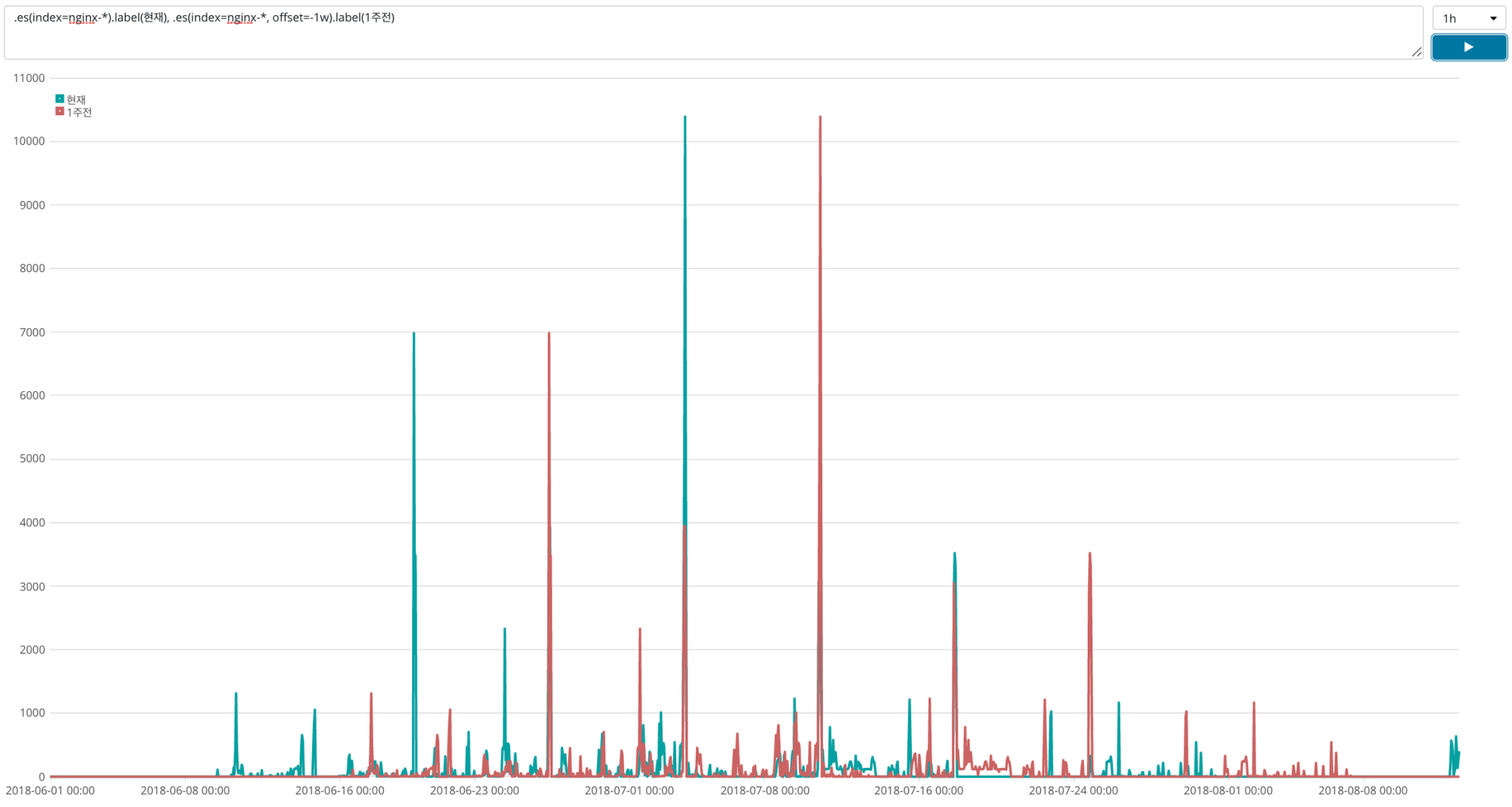
```
.es(index=nginx-*, q=-nginx.access.response_code:200).divide( .es(index=nginx-*) ).label(에러비율)
```

살짝 변경

에러율의 추이를 확인할 수 있다

Timelion - offset 👑

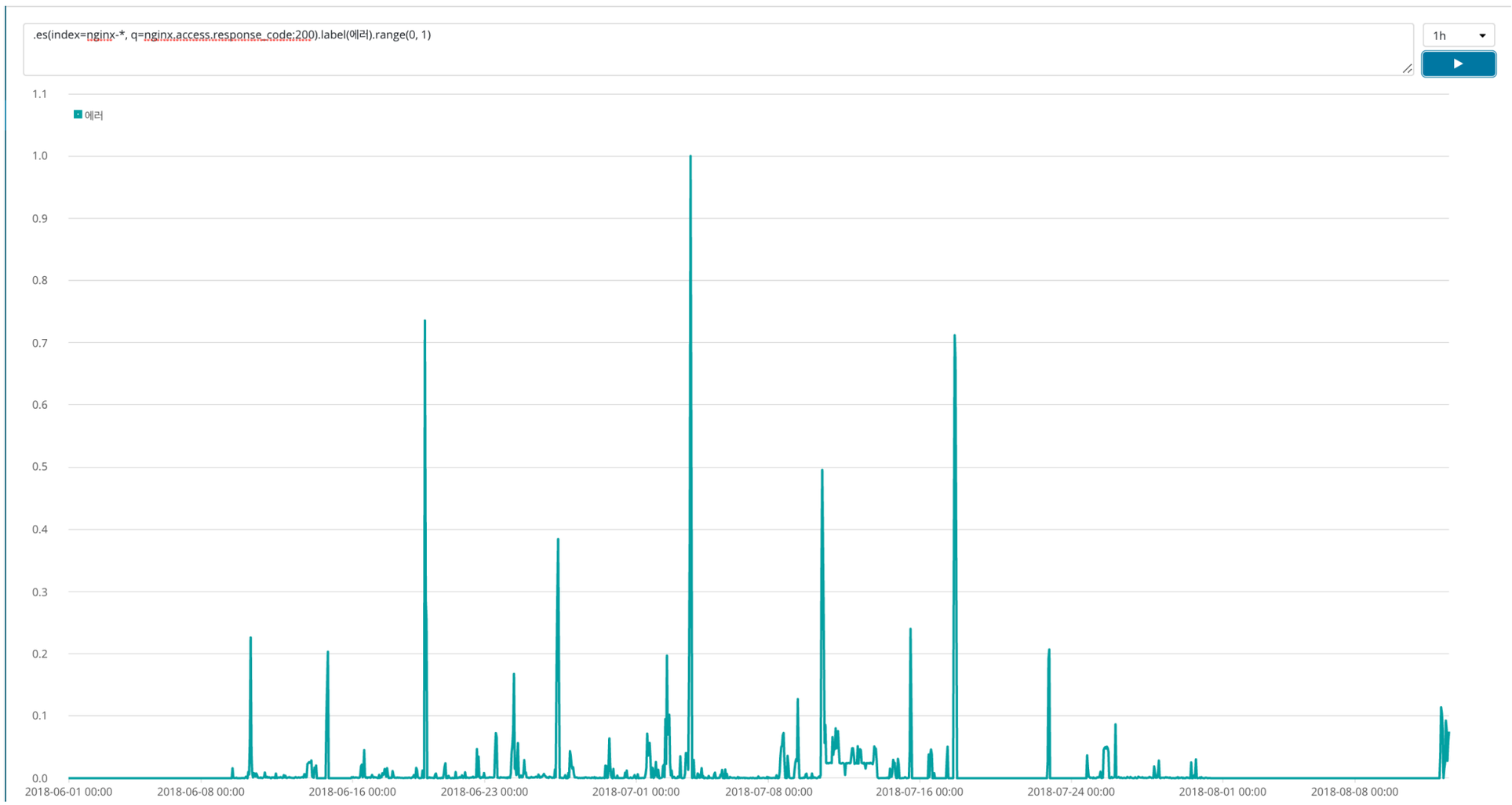
.es(index=nginx-*).label(현재) , .es(index=nginx-*, offset=-1w).label(1주전)



“현재 수치”와 **“1주일 전 수치”**를 **동일 선**에서 **확인 가능**

Timelion - range 👑

.es(index=nginx-*, q=nginx.access.response_code:200).label(에러).range(0, 1)

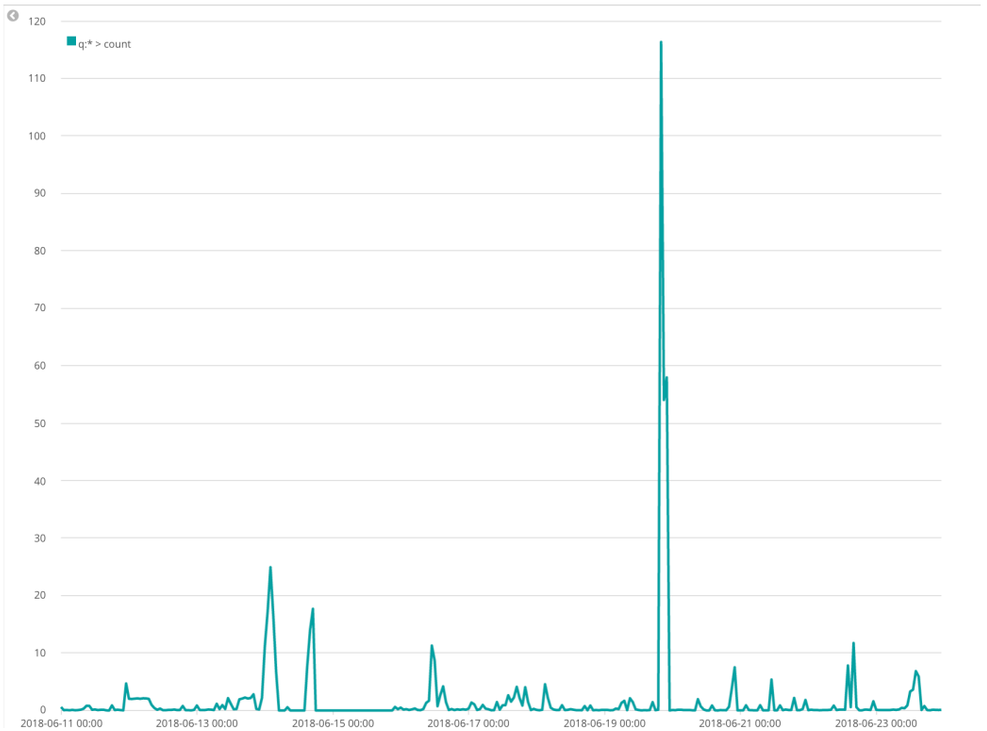


기간 내 에러 개수의 상대적 추이를 확인할 수 있다

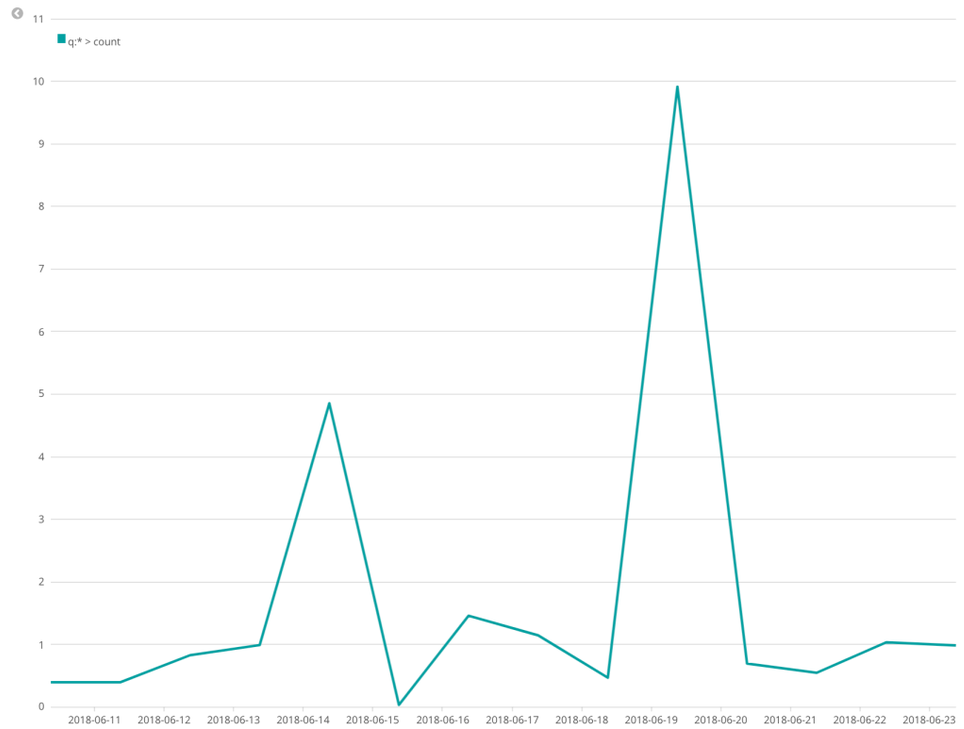
page 60 ~ 68 부분은 데이터 수집 문제로 2018년 6월 데이터를 사용합니다

Timelion - scale interval 👑

Interval : **1h**
Timelion : .es(index=nginx-*).scale_interval(1m)



Interval : **1d**
Timelion : .es(index=nginx-*).scale_interval(1m)



Interval이 다른 2개의 그래프에서 같은 시간 단위 동안의 접속수를 비교할 수 있다

Timelion - scale interval 👑

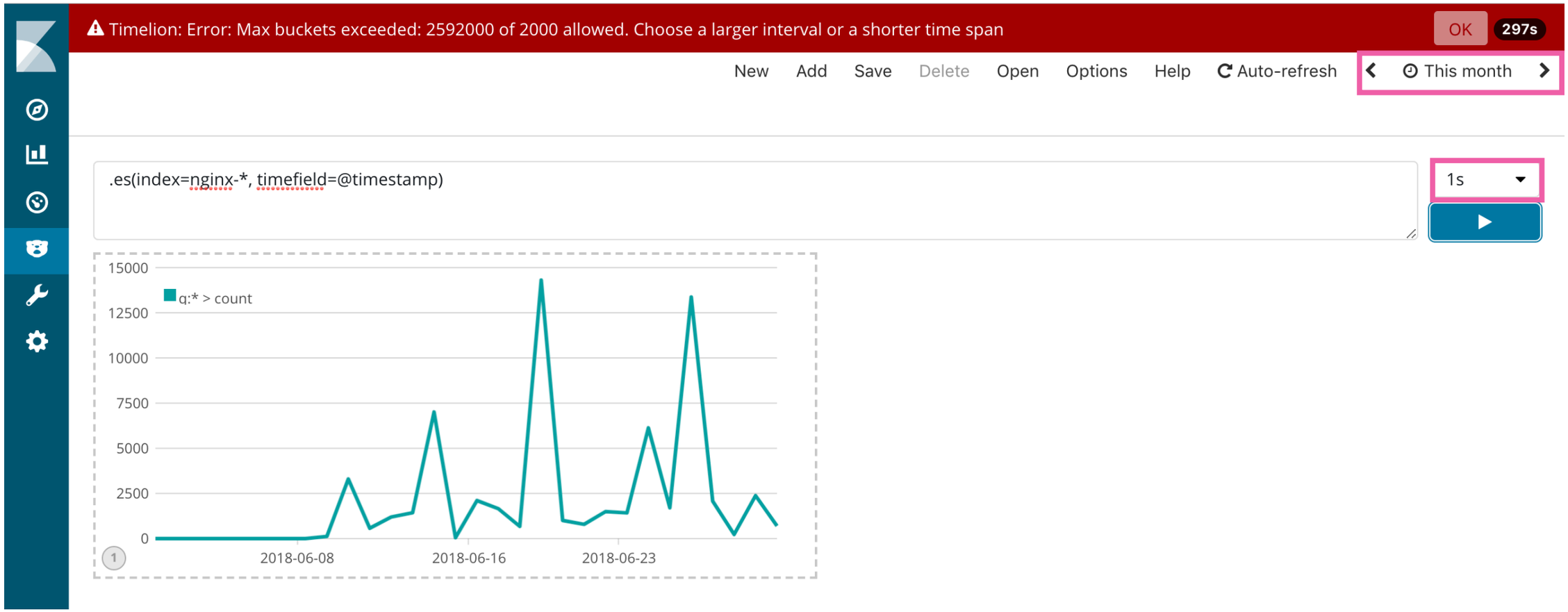
Interval : **1h**
Timelion : .es(index=nginx-*).scale_interval(1m)

Interval : **1d**
Timelion : .es(index=nginx-*).scale_interval(1m)

잠깐5, 이거 언제 쓰이지?

Interval이 다른 2개의 그래프에서 같은 시간 단위 동안의 접속수를 비교할 수 있다

Dashboard에서 사용하는 Timelion은 보통 Interval을 Auto로 한다 **심화**

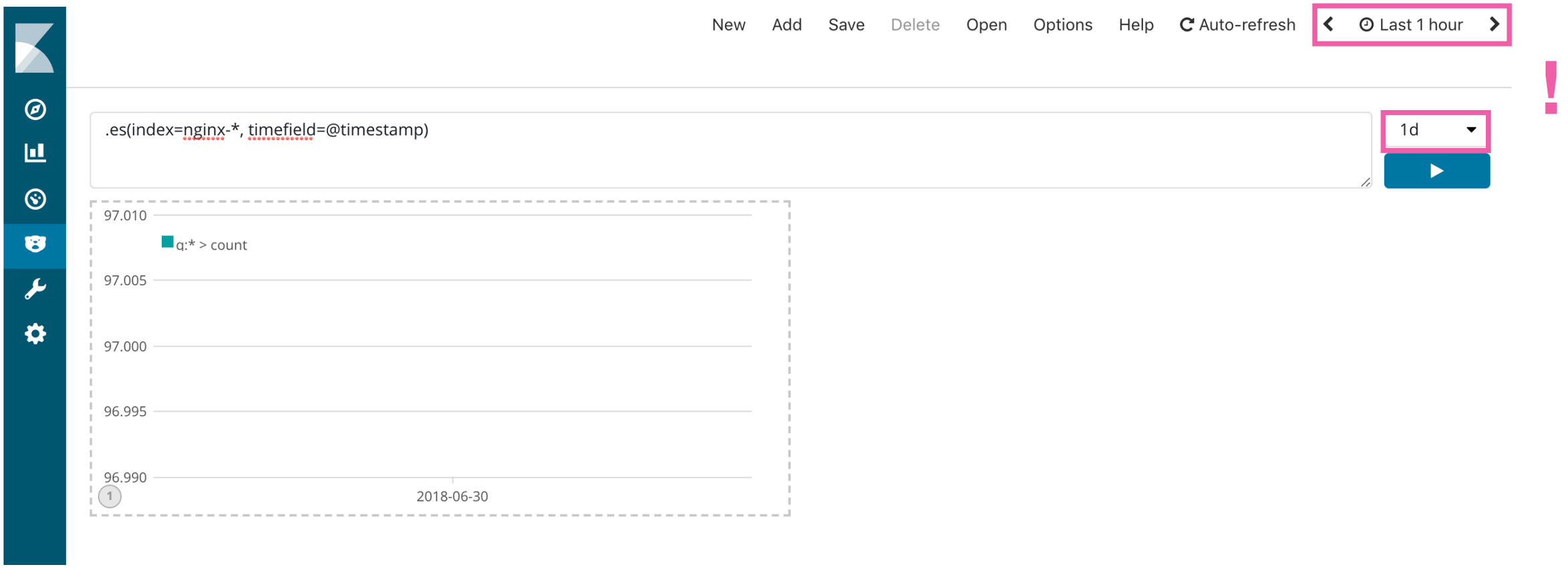


!

Time Picker 기간 대비 Interval을 너무 **작게** 설정하면

Max buckets exceeded error

Dashboard에서 사용하는 Timelion은 보통 Interval을 Auto로 한다 **심화**

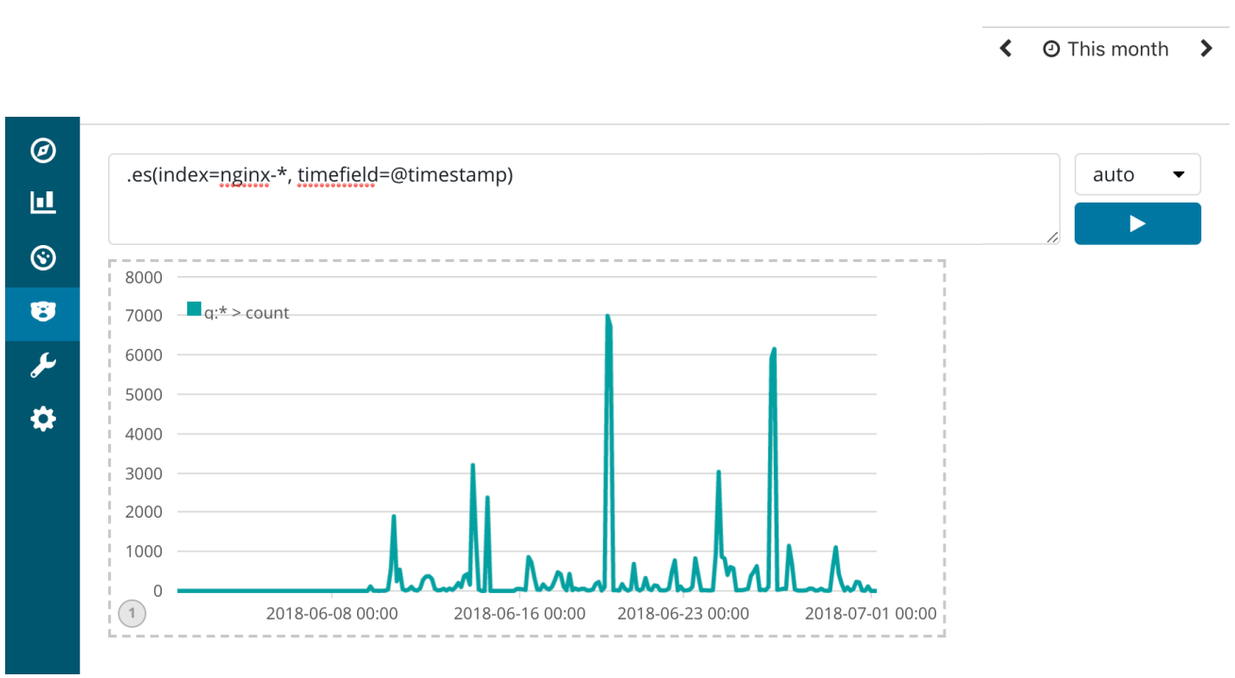


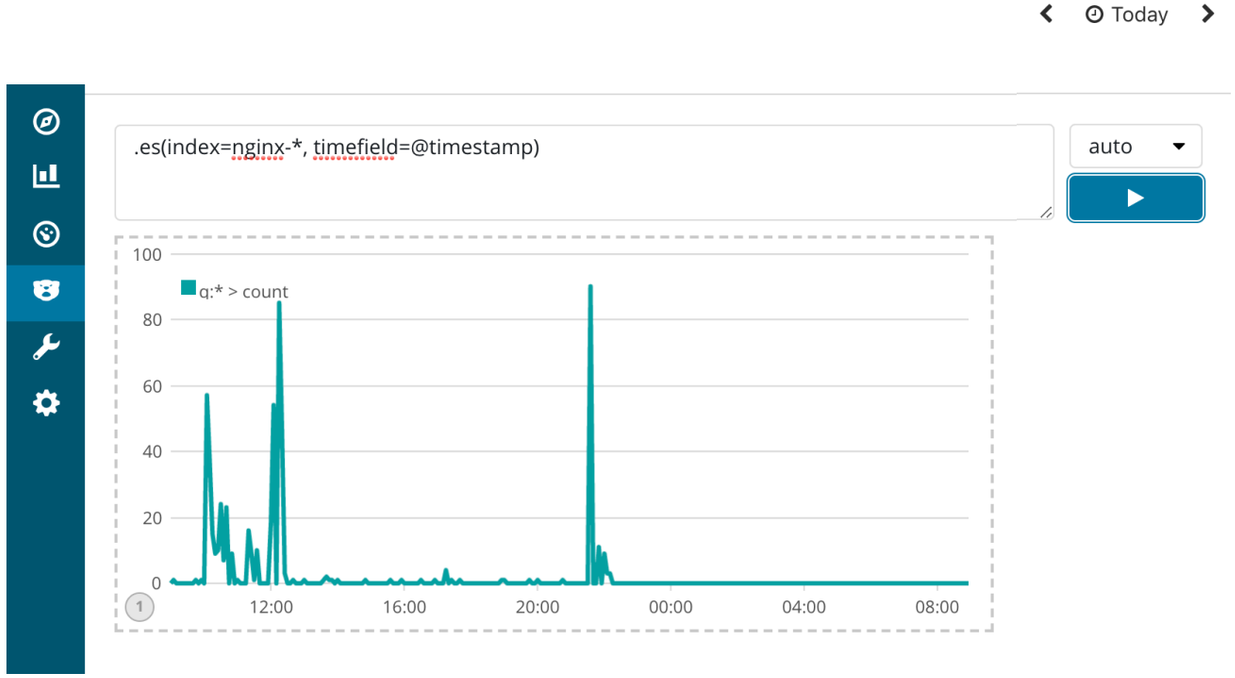
!

Time Picker 기간 대비 Interval을 너무 **크게** 설정하면

유의미한 결과를 볼 수 없다

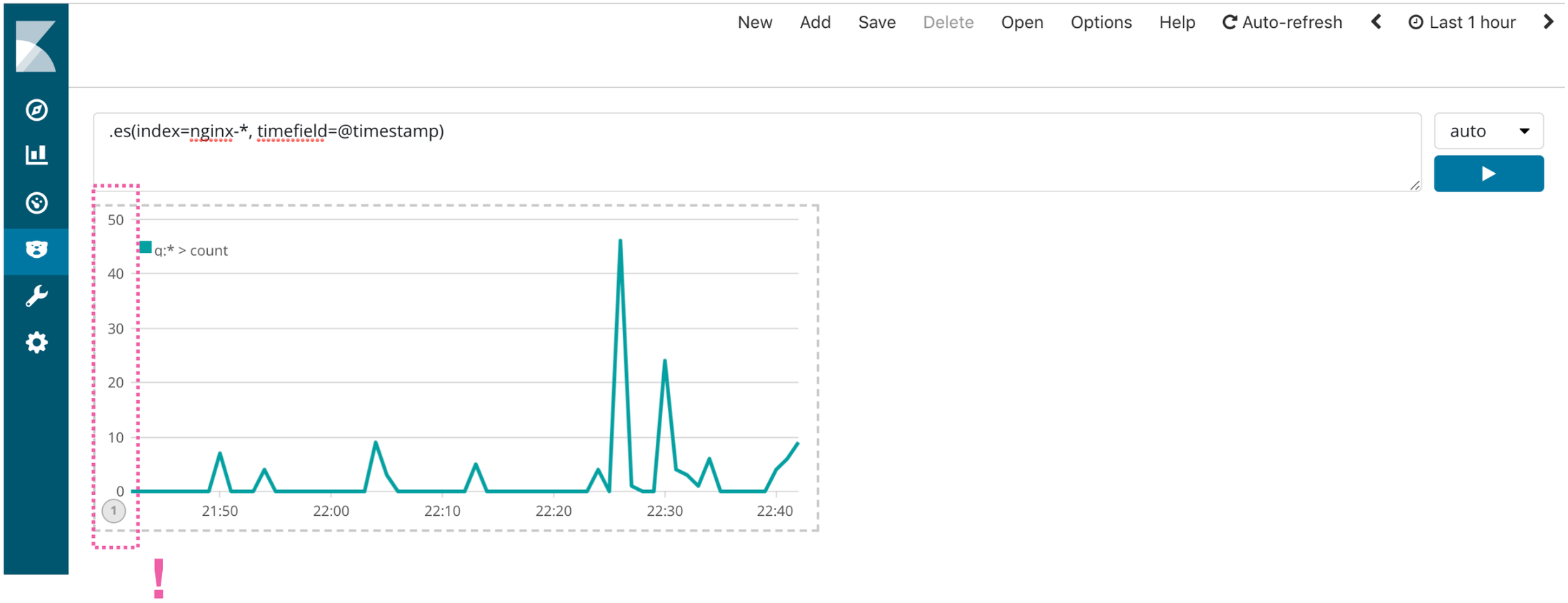
Interval을 Auto로 하면 Time Picker 기간에 맞게 유연하게 Interval 설정 가능 **심화**





Time Range를 **This month**로 하면 Interval이 커지고, **Today**로 하면 Interval이 작아진다

단, 그럴 경우 의도치 않은 다른 문제 발생 **심화**



분당(1 minute) document count를 모니터링 한다고 하자

단, 그럴 경우 의도치 않은 다른 문제 발생 **심화**

!

이 때 Time Range를 넓히면 Interval도 (auto이지만 실제로) **hour**, **day**로 커지고 표시되는 결과도 바뀐다

- document count per hour
- document count per day

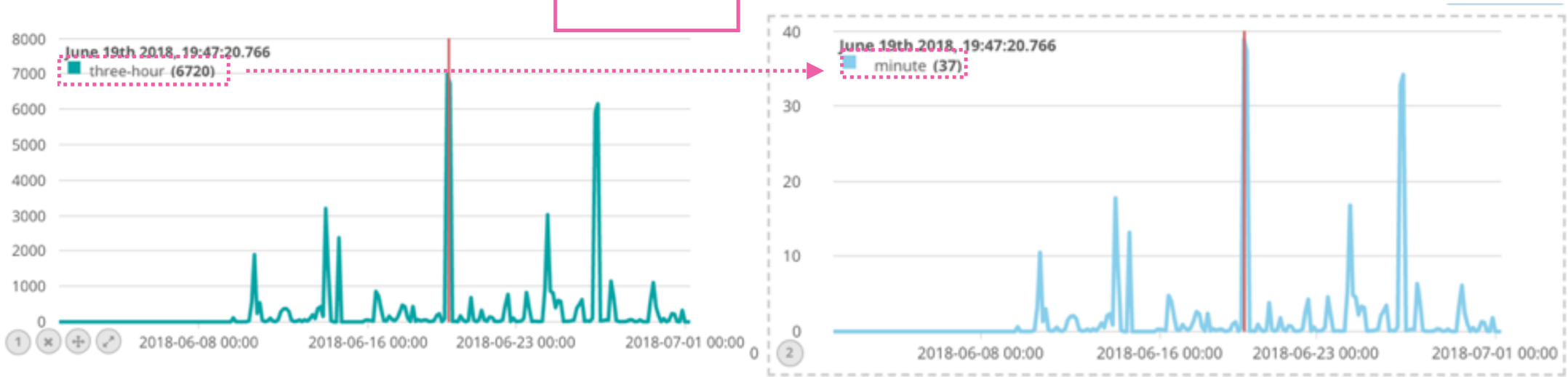
≠ document count per minute

즉, Time Range를 길게하면서도 Interval은 작게 유지하려 했는데
Time Range를 크게 한만큼 Interval도 커지는 결과 발생

↓ Interval이 바뀌어도 scale은 일정하게 유지할 수 없을까?

Interval이 변경되어도 Time Range가 15분이든, today든 this year든
scale을 유지하여 **분당 document count 결과**를 볼 수 없을까?

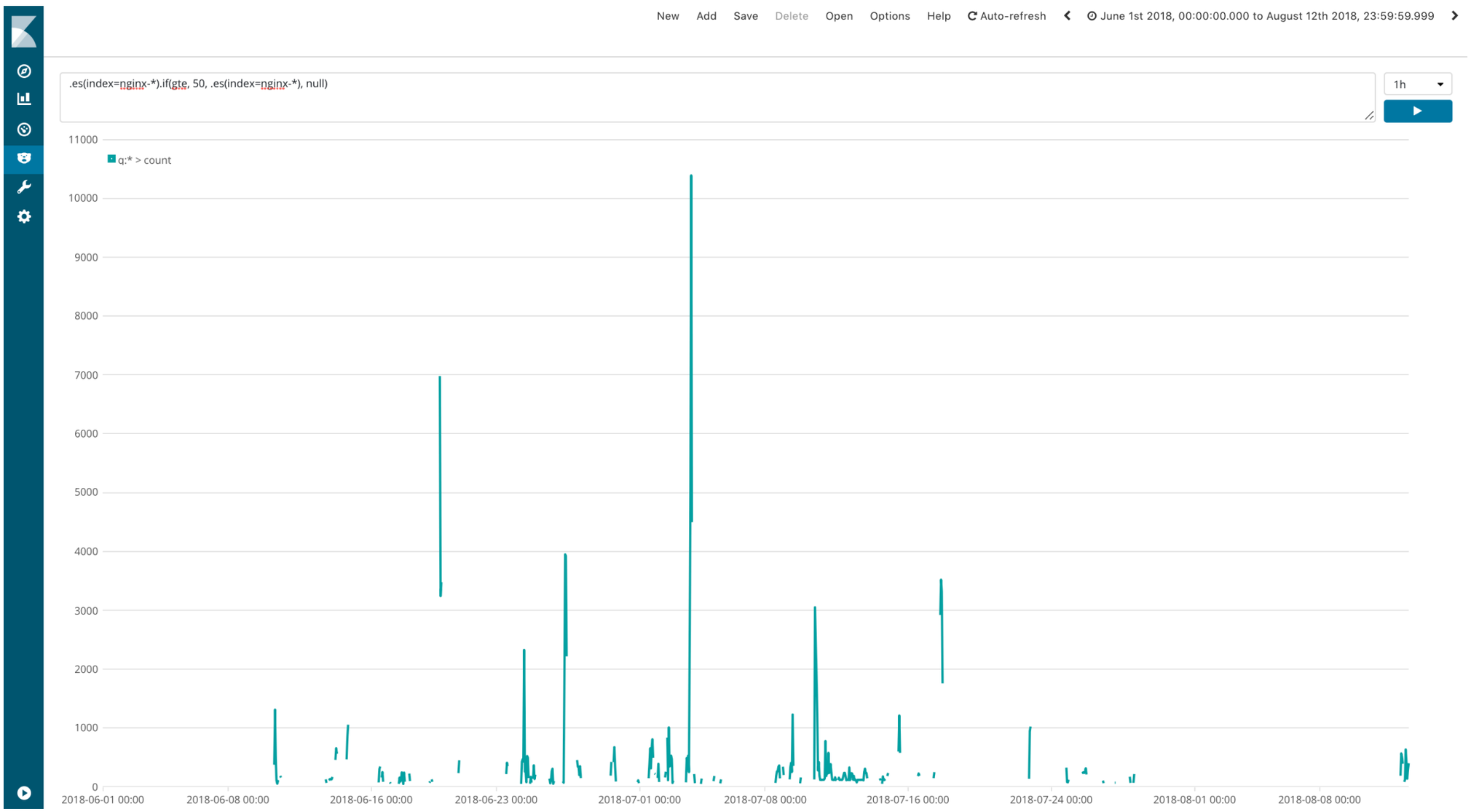
÷ 180



Time Range	This month	
expression	.es(index=nginx-*)	.es(index=nginx-*).scale_interval(1m)
Interval	**Auto (3h)**	
Scale	**3 hours**	**1 minute**
metrics	3시간 별 document count	(환산된) 1분 별 document count
해석	6720 : (Interval 동안) 누적 document count	37 : (Interval 동안) 누적 document count를 scaling한 결과

Timelion - if 👑

```
.es(index=nginx-*) .if( gte, 50, .es(index=nginx-*), null )
```



조건에 따른 다른 값 시각화 (문법 👑)

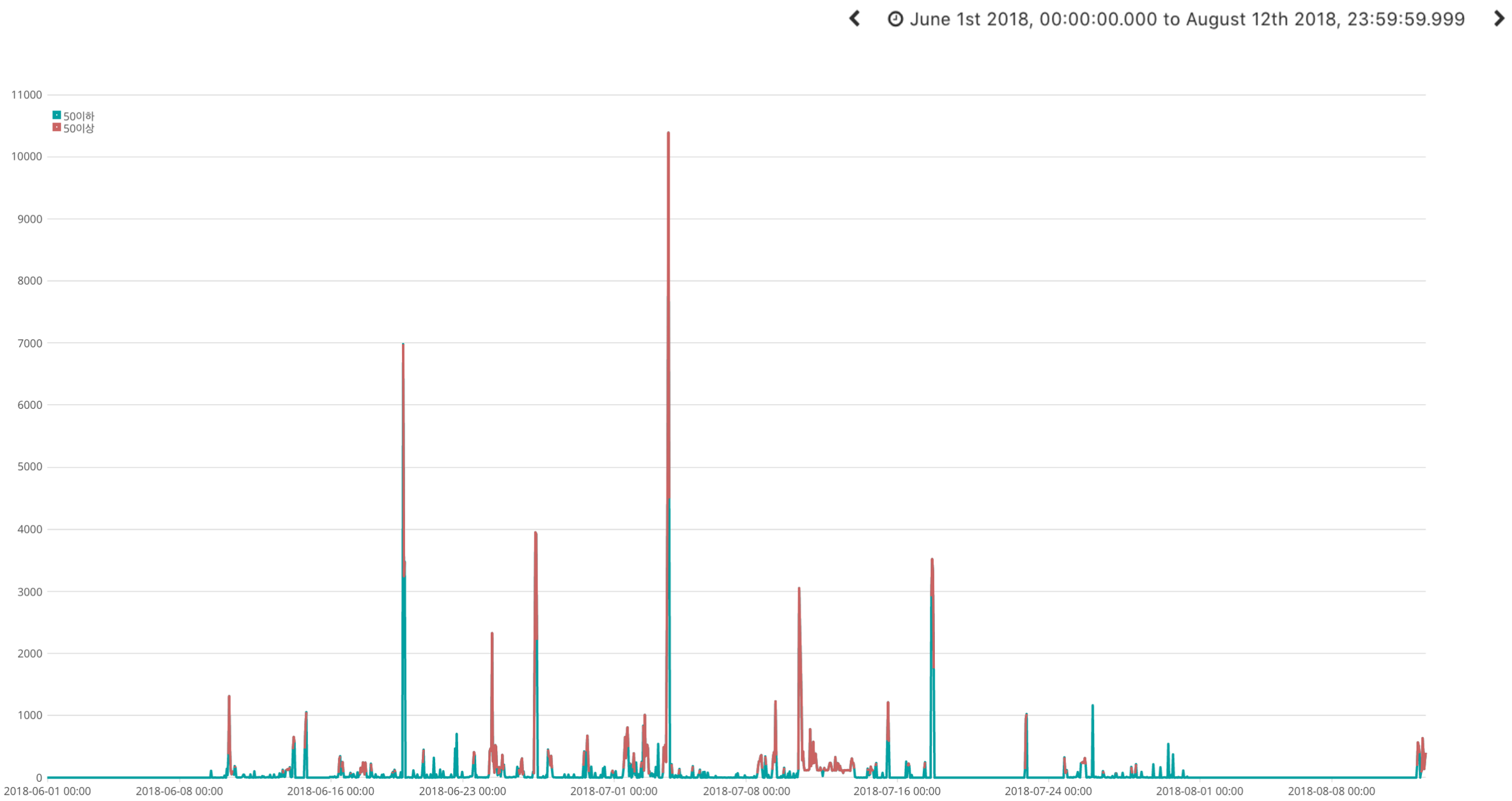
이 정도가 Timelion의 기본문법이며, 나머지는 👑를 참고해서 한 번씩 입력해보자. (수업 후에)

그리고, 다음 페이지에 나오는 몇 가지 예시를 보면서 Timelion을 마무리 하자.

이 때, Timelion 기능을 자신의 문제에 어떻게 응용/적용할 수 있을 지 생각해보자.

예제 15) Timelion

nginx-* index의 시간당(1h) 접속수가 50 이상인 구간과 50 이하인 구간을 다르게 표시하자



기본적인 시각화는 다 배웠다.

남겨둔 **Aggregation**을 마저 배우고

Kibana에서 제공하는 **chart/aggregation**을 다 쓸 수 있도록 하자

Aggregation - Pipeline

Kibana에서 사용할 수 있는 Aggregation

= ~~Bucket + Metrics~~ + Pipeline

(… 지난 주에 배웠고)

Pipeline Aggregation 👑

Pipeline Aggregations

① 특징

Pipeline aggregations work on the outputs produced from other aggregations rather than from document sets, adding information to the output tree. There are many different types of pipeline aggregation, each computing different information from other aggregations, but these types can be broken down into two families:

② 종류

Parent

A family of pipeline aggregations that is provided with the output of its parent aggregation and is able to compute new buckets or new aggregations to add to existing buckets.

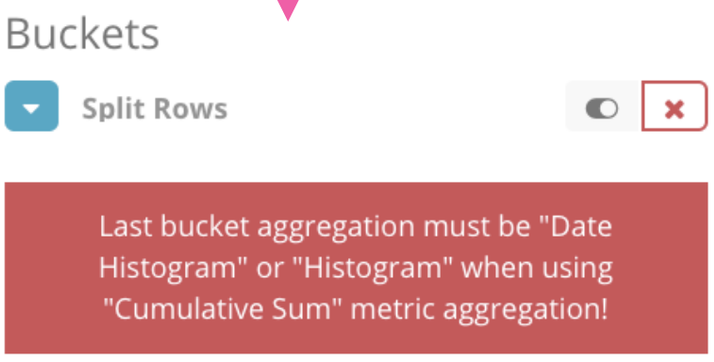
Sibling

Pipeline aggregations that are provided with the output of a sibling aggregation and are able to compute a new aggregation which will be at the same level as the sibling aggregation.

Aggregation - Parent Pipeline

종류	상세
Derivative	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, 연속한 Bucket 간의 차이를 구함
Cumulative Sum	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Bucket 값들의 누적합을 구함
Moving Average	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, {n개} Bucket 간의 평균을 구함
Serial Differencing	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, {n번째 이전} Bucket 과의 차이를 구함

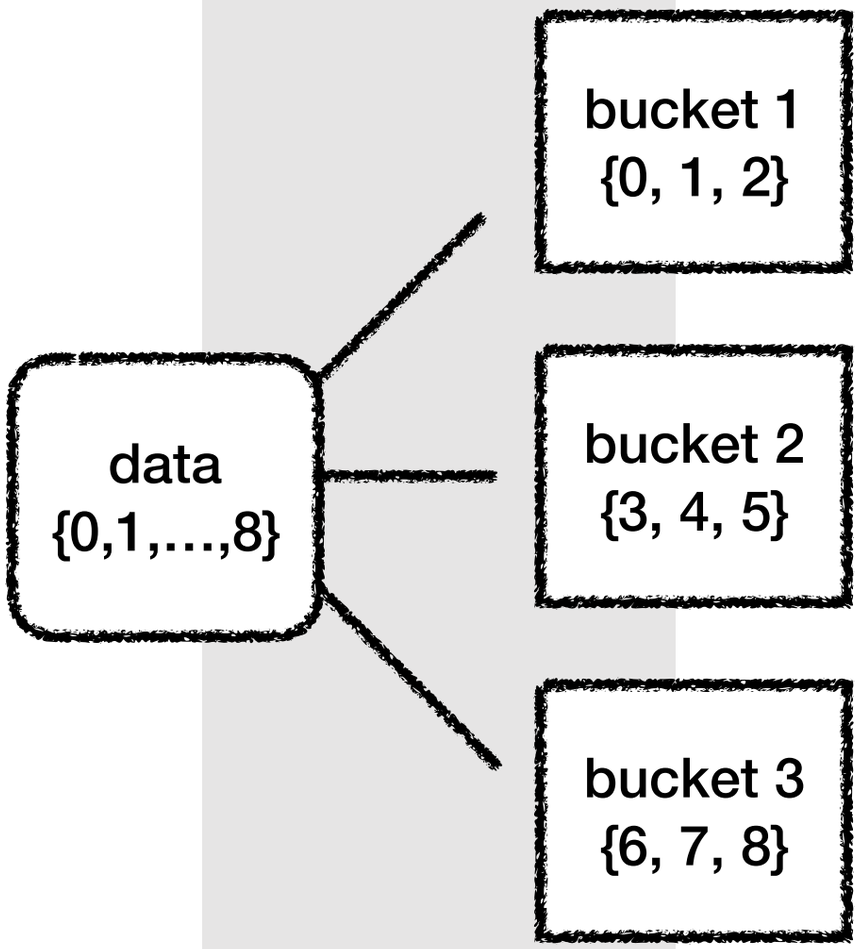
(Date Histogram 또는 Histogram 만 지원)

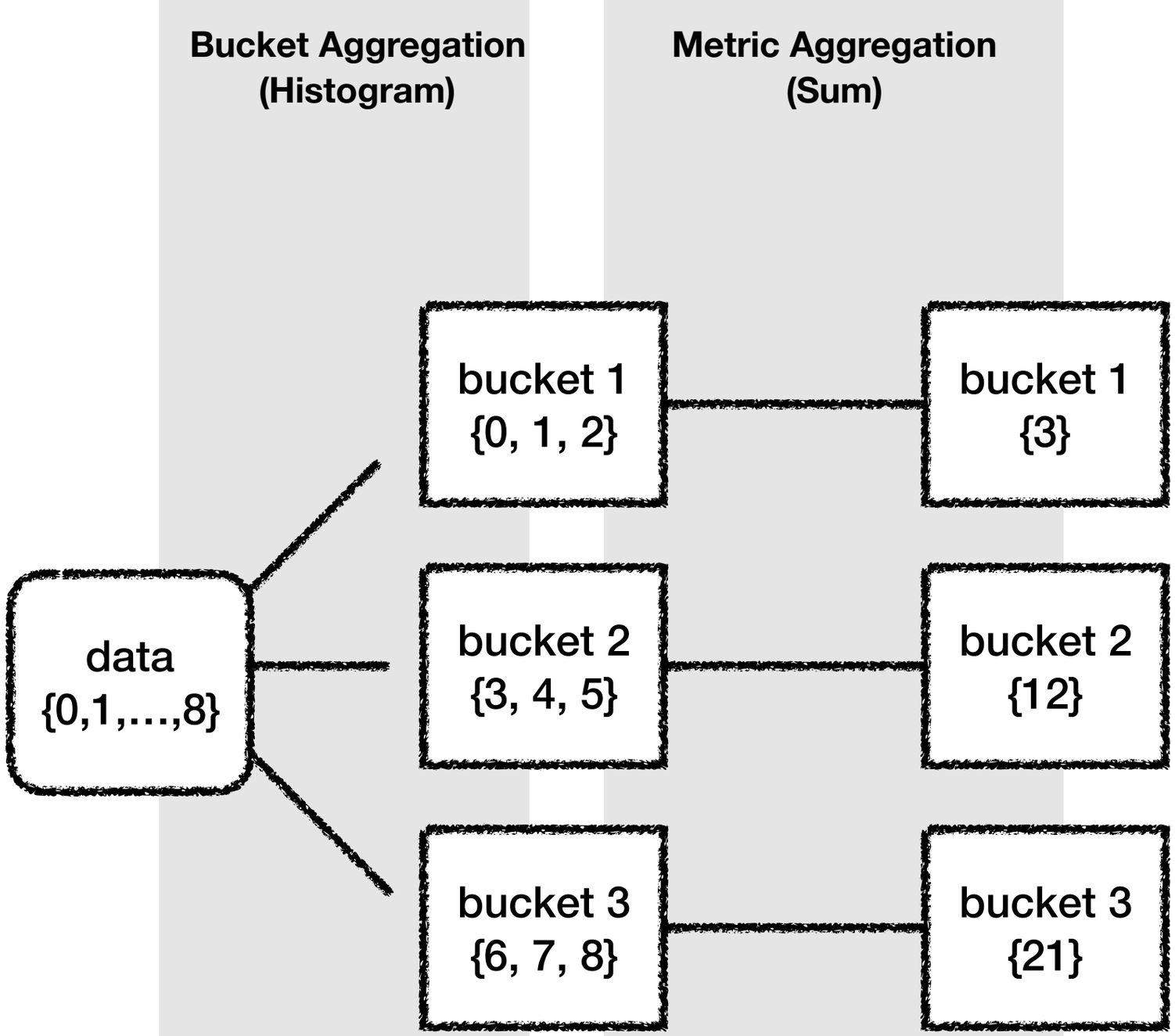


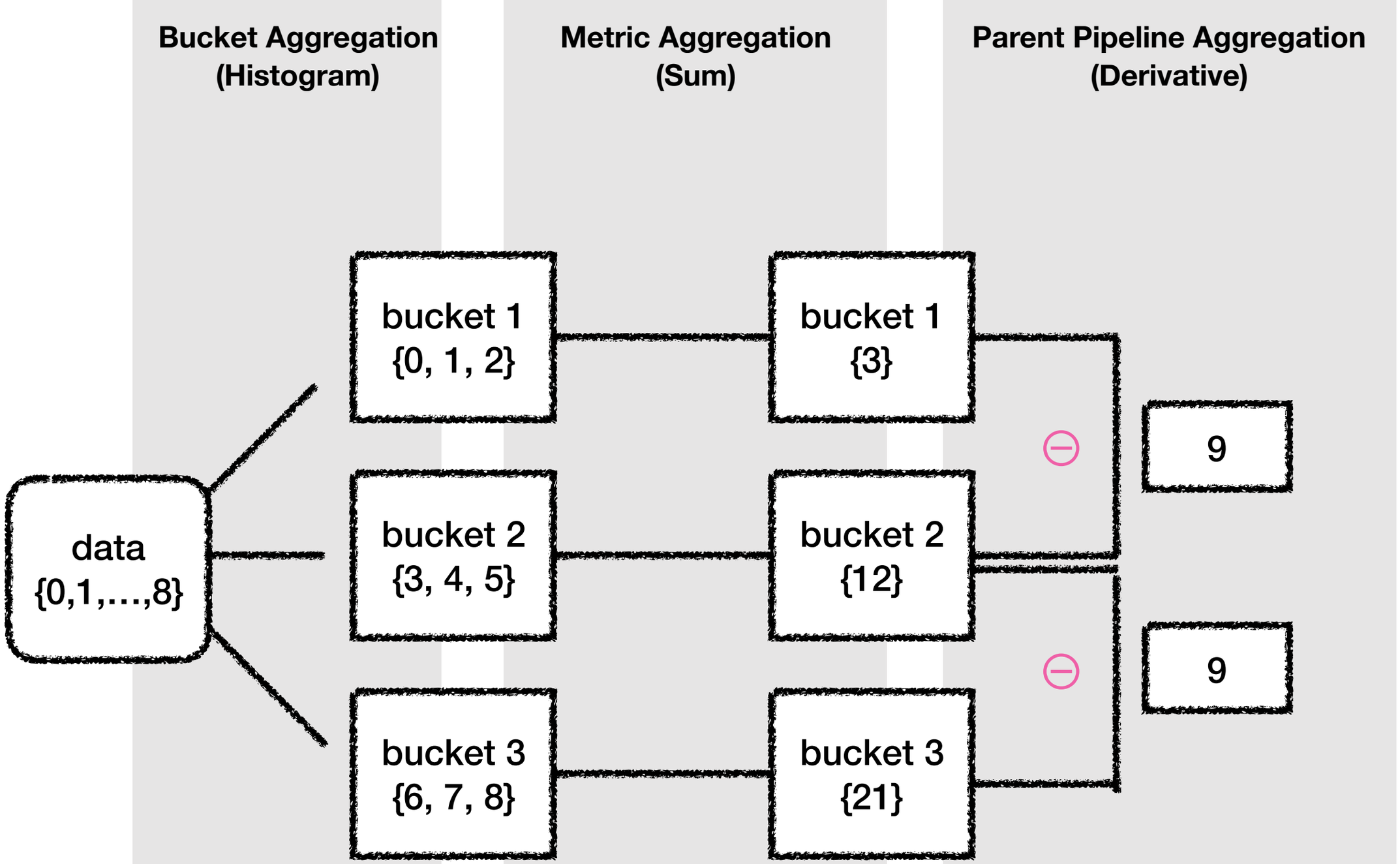
다음과 같은 데이터가 있다고 하자

{“data” : 0}	{“data” : 1}	{“data” : 2}
{“data” : 3}	{“data” : 4}	{“data” : 5}
{“data” : 6}	{“data” : 7}	{“data” : 8}

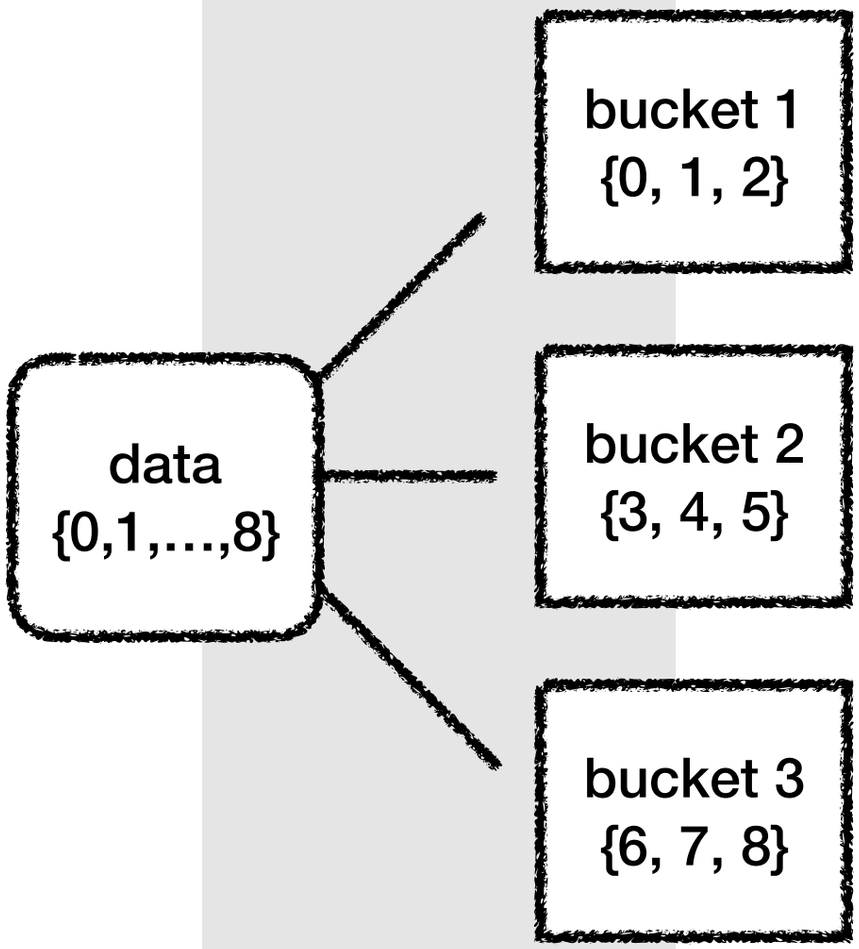
Bucket Aggregation (Histogram)

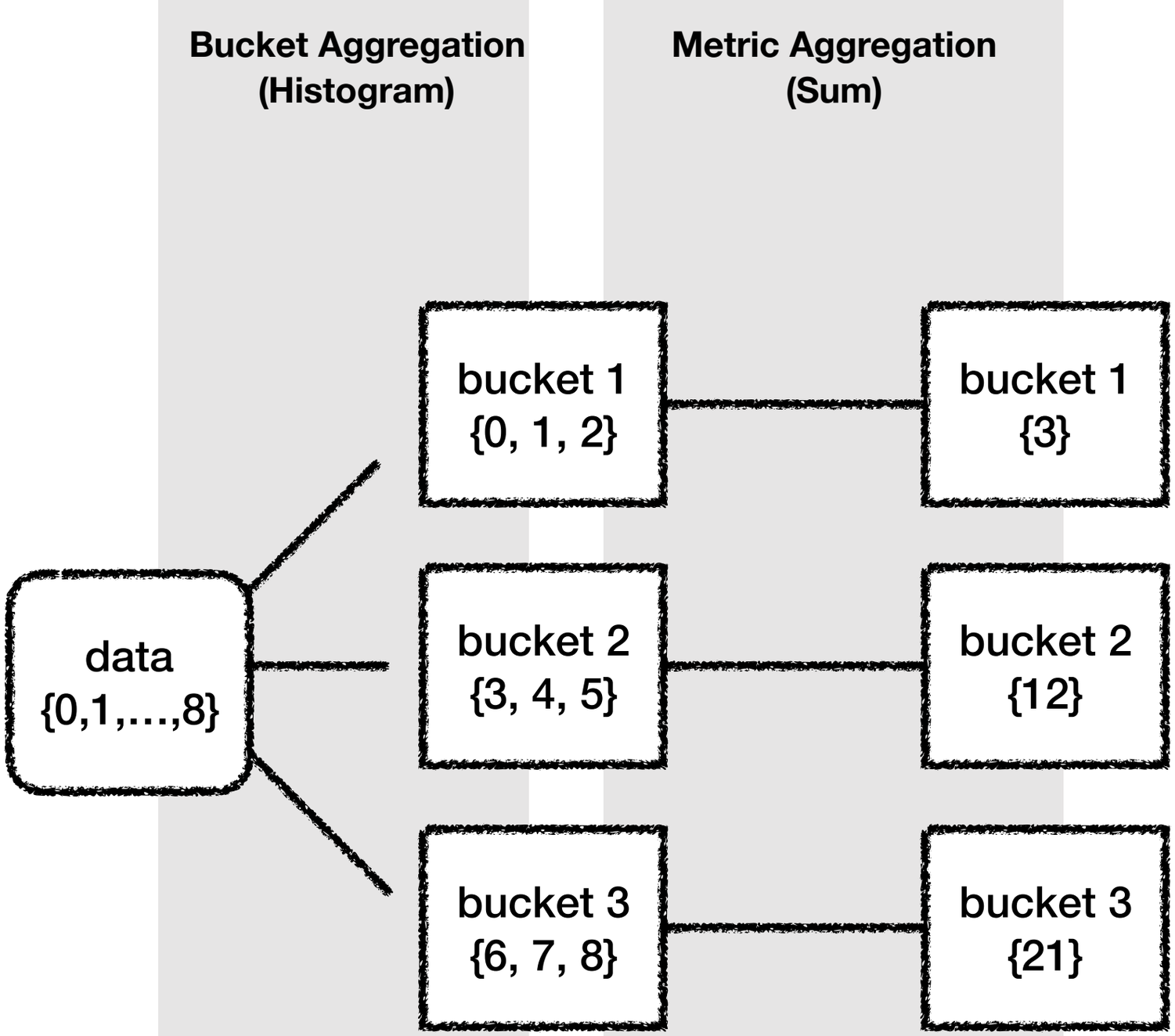


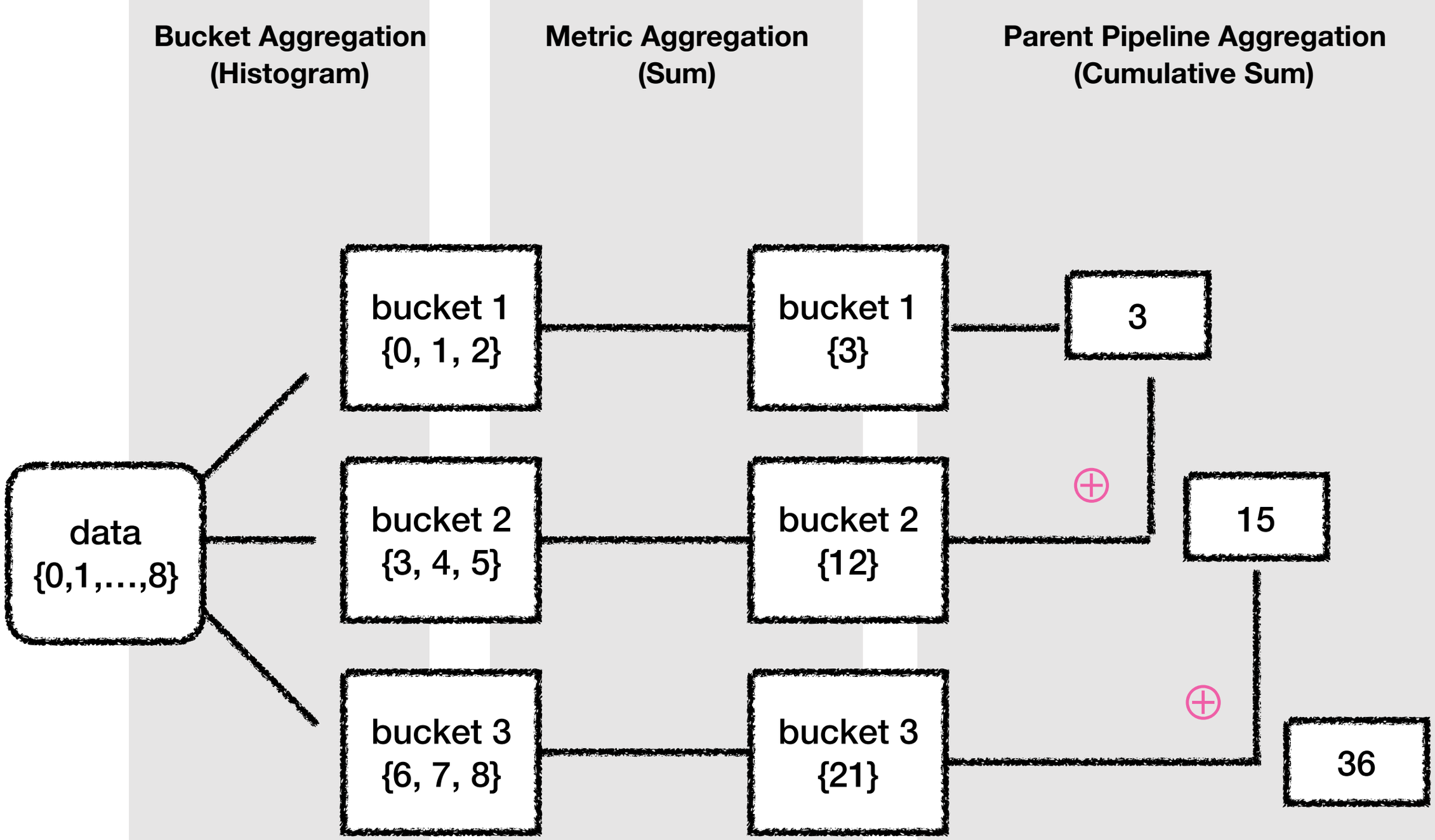

Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}


Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
Parent Pipeline Aggregation
(Derivative)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}
⊖
⊖
9
9

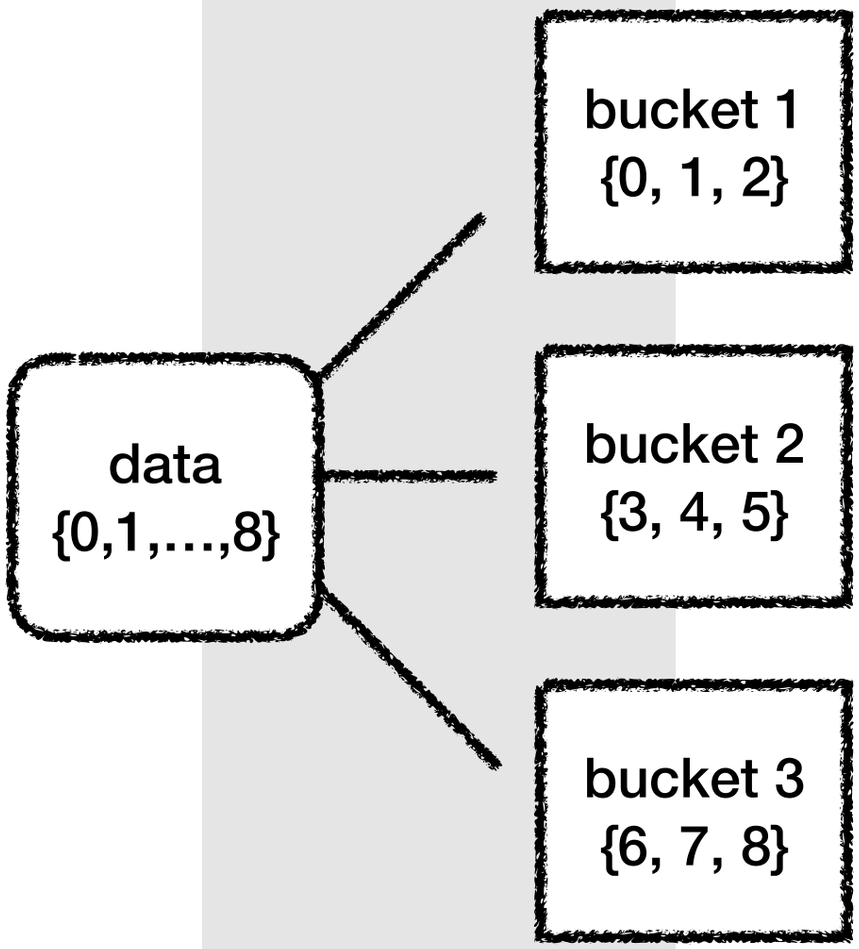
Bucket Aggregation (Histogram)

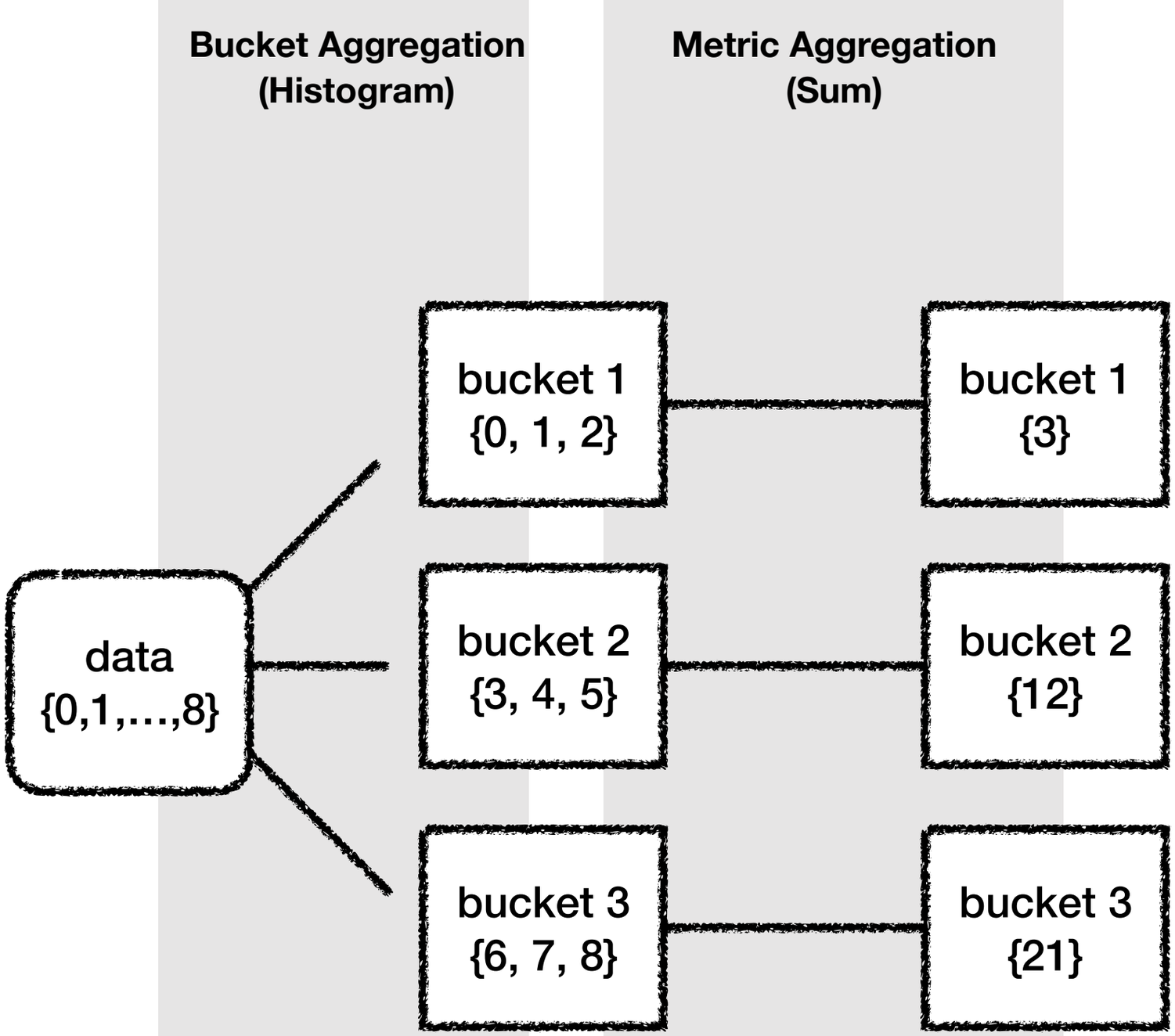


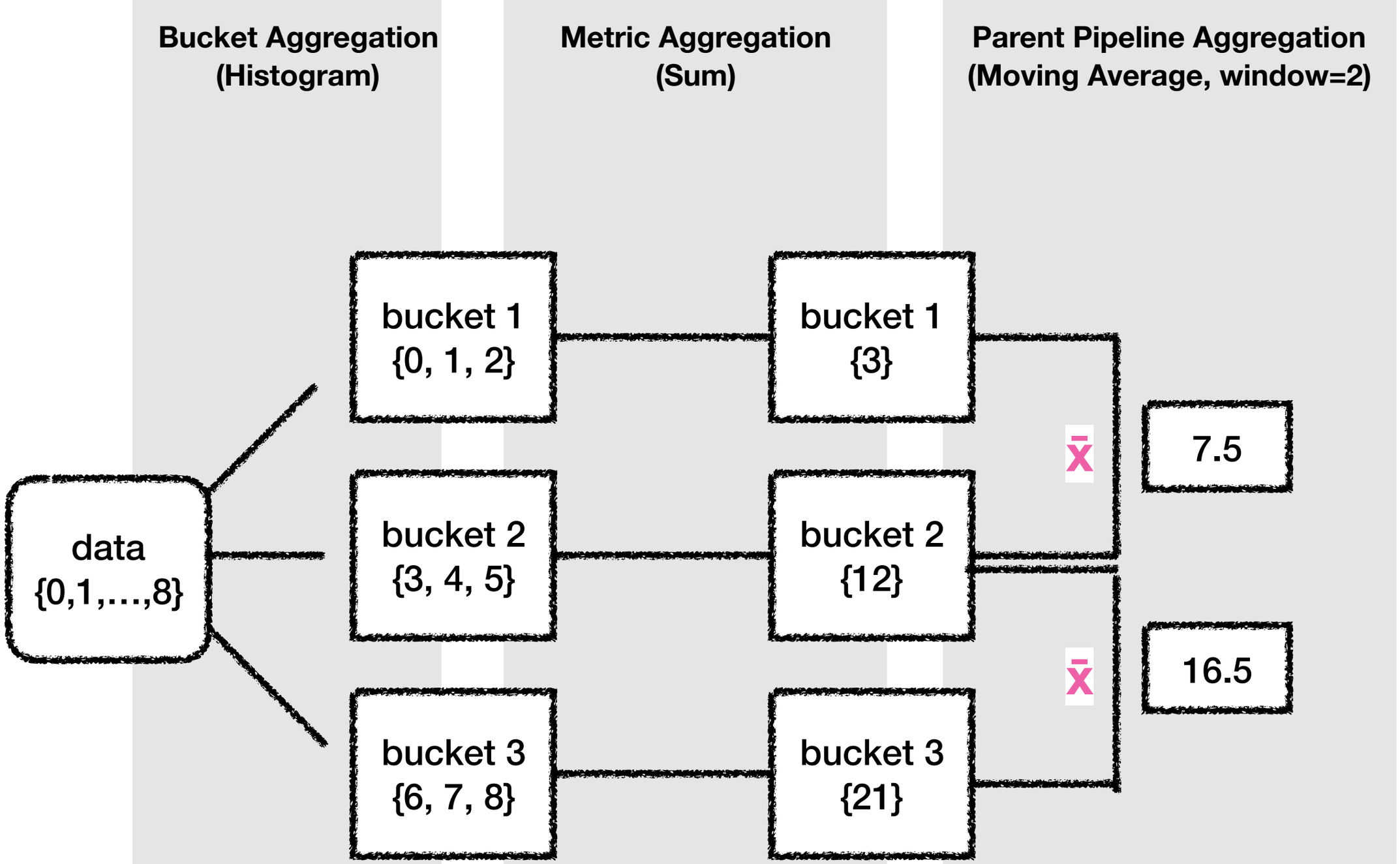

Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}


Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
Parent Pipeline Aggregation
(Cumulative Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}
3
15
36

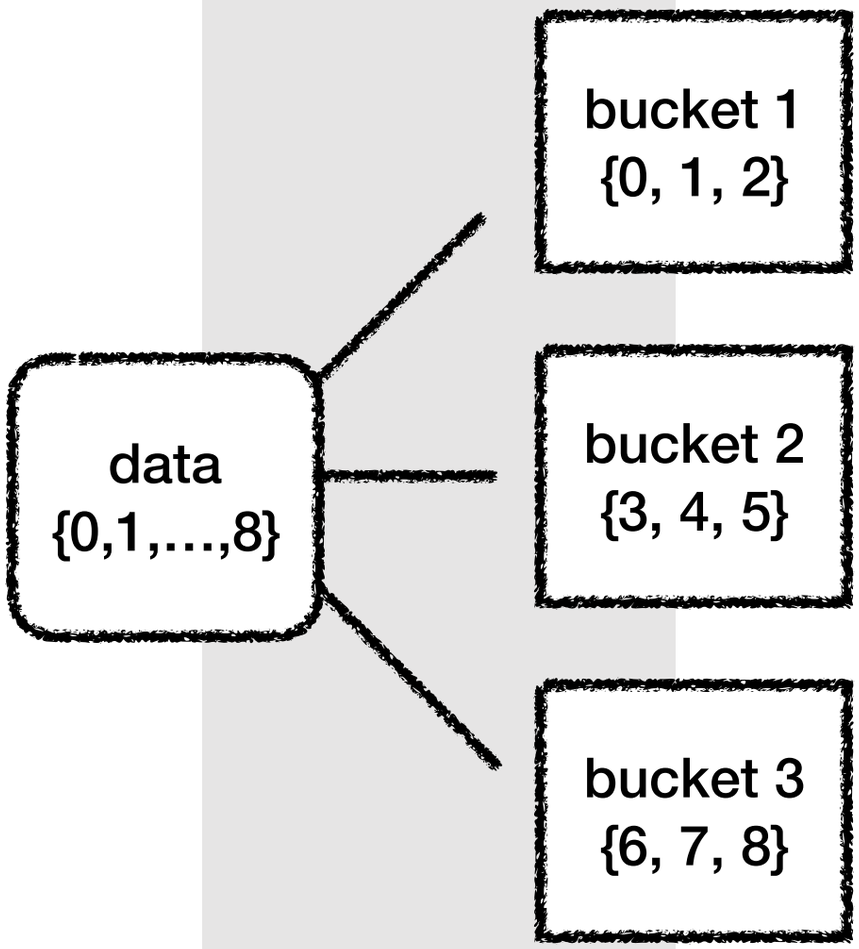
Bucket Aggregation (Histogram)

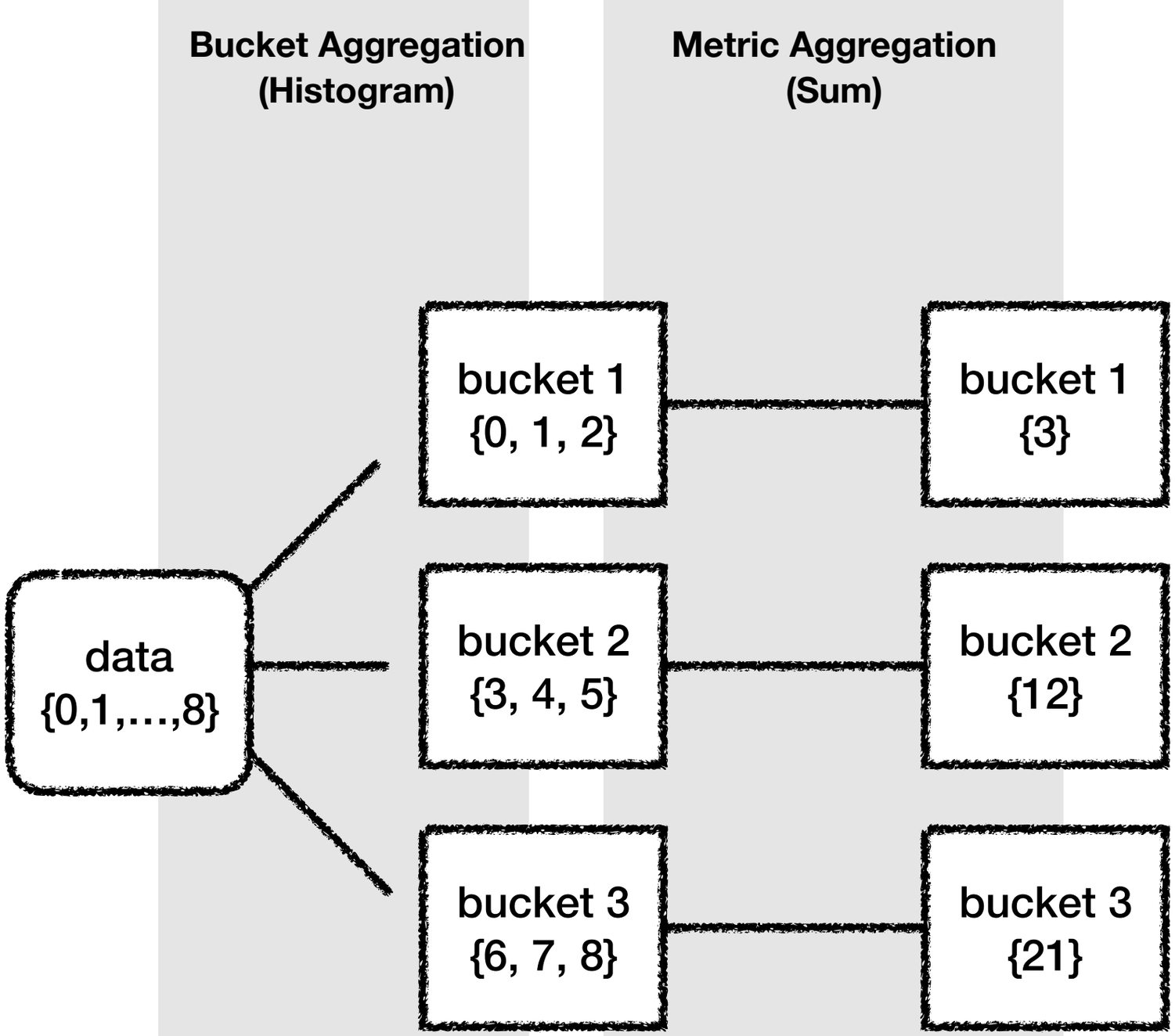


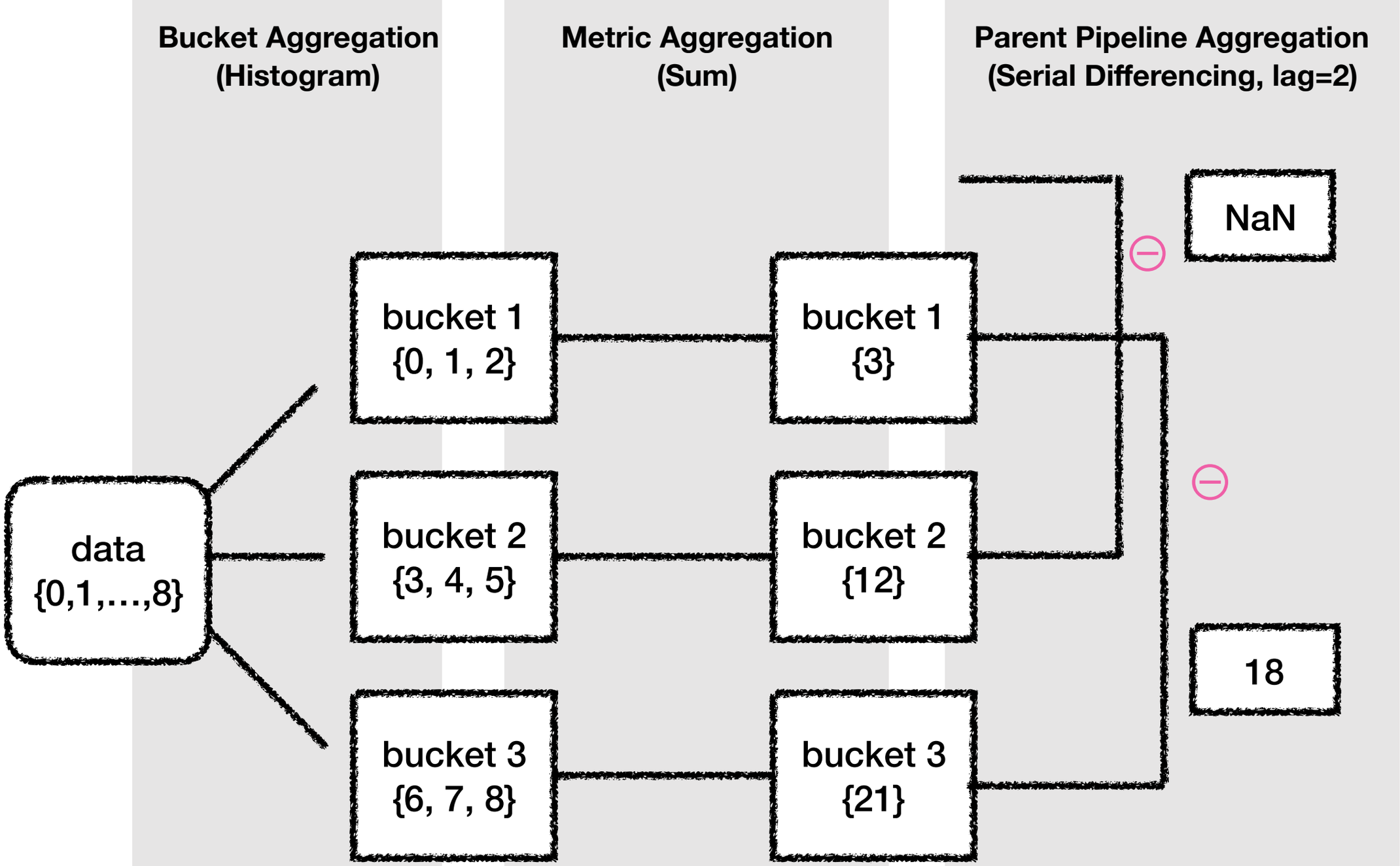

Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}


Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
Parent Pipeline Aggregation
(Moving Average, window=2)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}
x̄
x̄
7.5
16.5

Bucket Aggregation (Histogram)




Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}


Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
Parent Pipeline Aggregation
(Serial Differencing, lag=2)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}
NaN
18

실제 Kibana에서 **Parent** Pipeline Aggregation을 어떻게 사용하는지 보자

Data Table Object

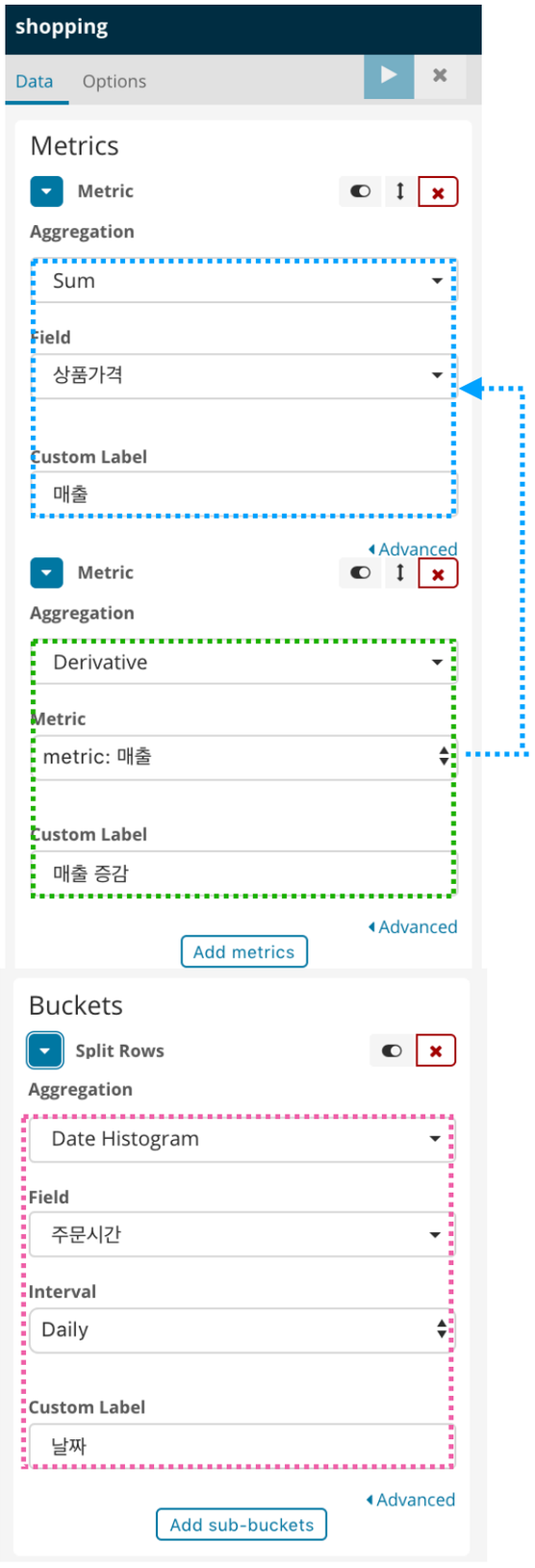
날짜	매출	매출 증감
2018-01-01	1,845,000	-
2018-01-02	1,492,000	-353,000
2018-01-03	1,678,000	186,000
2018-01-04	1,878,000	200,000
2018-01-05	1,479,000	-399,000
2018-01-06	1,548,000	69,000
2018-01-07	1,715,000	167,000
2018-01-08	1,718,000	3,000
2018-01-09	1,562,000	-156,000
2018-01-10	1,424,000	-138,000

1,845,000 -1,492,000

조건

- **shopping** index 중에서
- “주문시간” field 기준 **this year** documents의
- “**주문시간**” field로 일별(daily) bucket을 만들고 — **일별**
- “**상품가격**” field의 **합**과 — **매출**
- "**일별 상품가격 field의 합**”의 **증감** — **매출 증감**

Data Table Configuration



Data Table Object

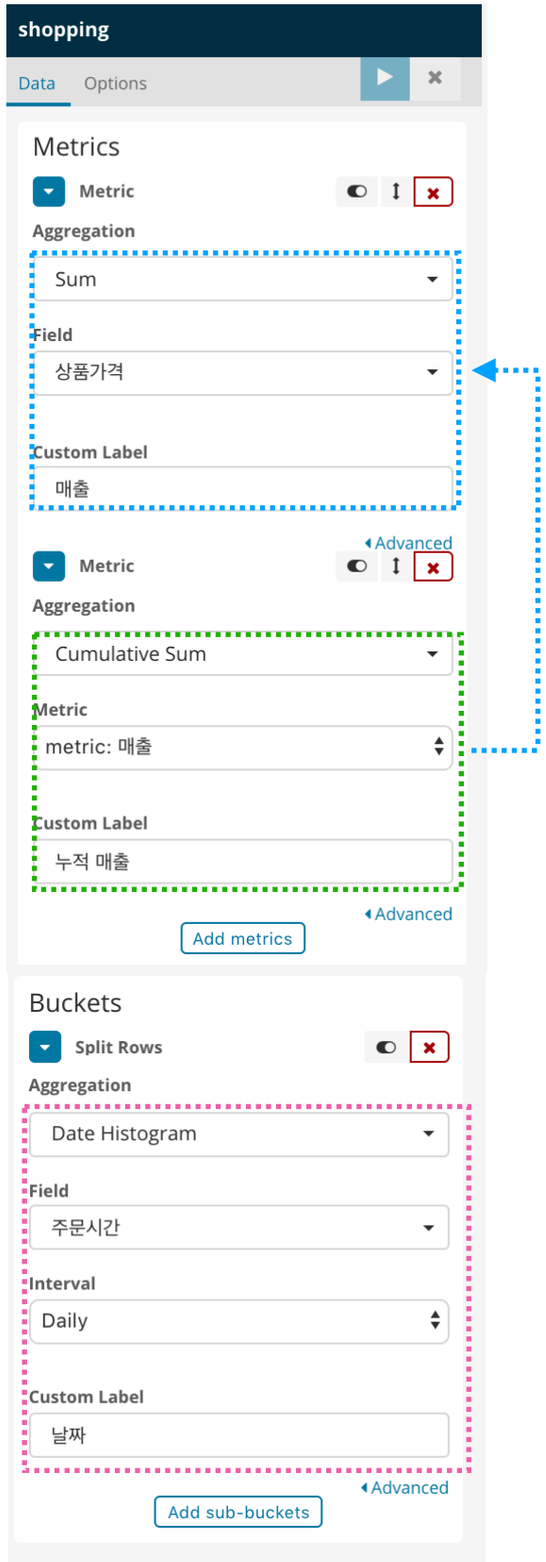
날짜	매출	누적 매출
2018-01-01	1,845,000	1,845,000
2018-01-02	1,492,000	3,337,000
2018-01-03	1,678,000	5,015,000
2018-01-04	1,878,000	6,893,000
2018-01-05	1,479,000	8,372,000
2018-01-06	1,548,000	9,920,000
2018-01-07	1,715,000	11,635,000
2018-01-08	1,718,000	13,353,000
2018-01-09	1,562,000	14,915,000
2018-01-10	1,424,000	16,339,000
2018-01-11	1,596,000	17,935,000

1,845,000 + 1,492,000

조건

- **shopping** index 중에서
- “주문시간” field 기준 **this year** documents의
- “**주문시간**” field로 일별(daily) bucket을 만들고 — **일별**
- “**상품가격**” field의 **합**과 — **매출**
- "**일별 상품가격 field의 합**”의 **누적합** — **누적 매출**

Data Table Configuration



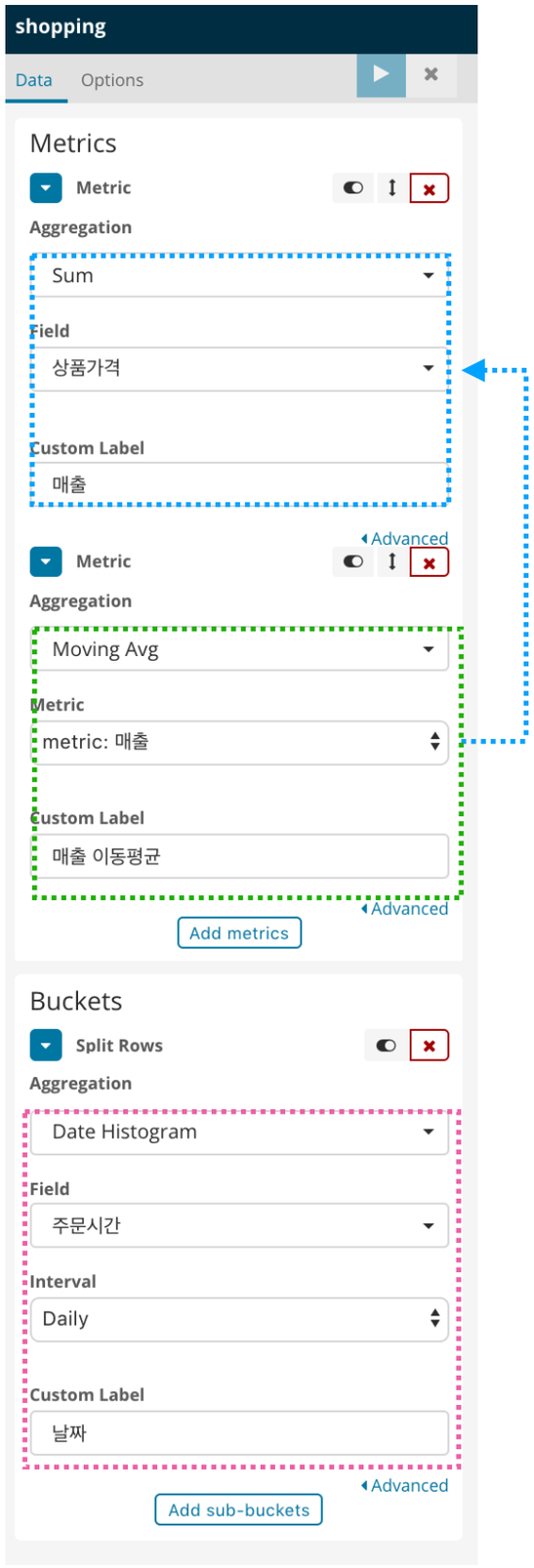
Data Table Object

날짜	매출	매출 이동평균
2018-01-01	1,845,000	-
2018-01-02	1,492,000	1,845,000
2018-01-03	1,678,000	1,668,500
2018-01-04	1,878,000	1,671,666.667
2018-01-05	1,479,000	1,723,250
2018-01-06	1,548,000	1,674,400
2018-01-07	1,715,000	1,615,000
2018-01-08	1,718,000	1,659,600
2018-01-09	1,562,000	1,667,600
2018-01-10	1,424,000	1,604,400

조건

- **shopping** index 중에서
- “주문시간” field 기준 **this year** documents의
- “**주문시간**” field로 일별(daily) bucket을 만들고 — **일별**
- “**상품가격**” field의 **합**과 — **매출**
- "**일별 상품가격 field의 합**”의 (5일) **이동평균** — **매출 이동평균**

Data Table Configuration



심화

잠깐6, 이동평균의 window size를 변경하고 싶다면?

Data Table Object 심화

날짜	매출	매출 이동평균 (5일)	매출 이동평균 (3일)
2018-01-01	1,845,000	-	-
2018-01-02	1,492,000	1,845,000	1,845,000
2018-01-03	1,678,000	1,668,500	1,668,500
2018-01-04	1,878,000	1,671,666.667	1,671,666.667
2018-01-05	1,479,000	1,723,250	1,682,666.667
2018-01-06	1,548,000	1,674,400	1,678,333.333
2018-01-07	1,715,000	1,615,000	1,635,000
2018-01-08	1,718,000	1,659,600	1,580,666.667
2018-01-09	1,562,000	1,667,600	1,660,333.333
2018-01-10	1,424,000	1,604,400	1,665,000

window : 3

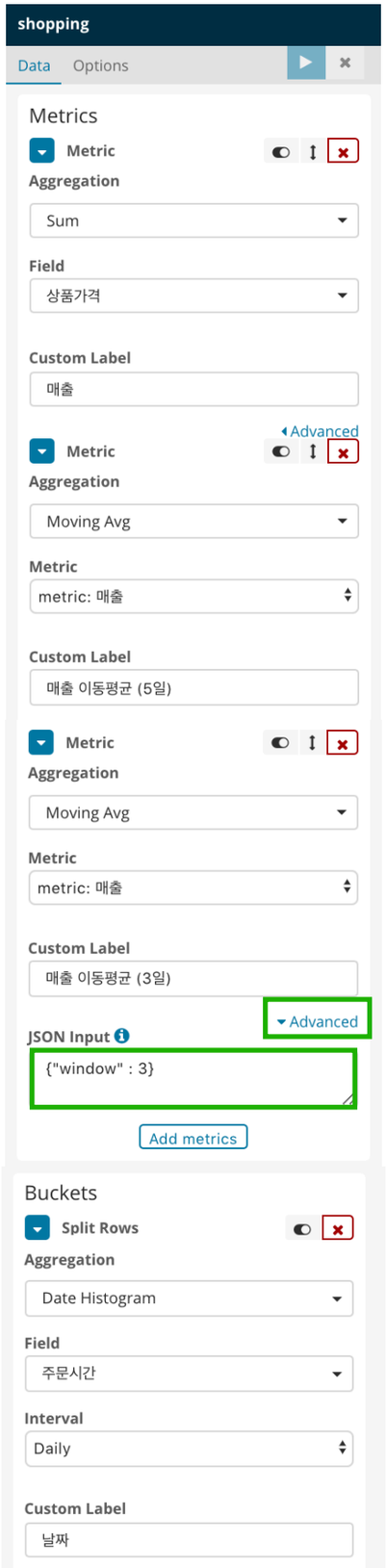
window : 5

조건

- **shopping** index 중에서
- “주문시간” field 기준 **this year** documents의
- “**주문시간**” field로 일별(daily) bucket을 만들고 — **일별**
- “**상품가격**” field의 **합**과 — **매출**
- "**일별 상품가격 field의 합**”의 (5일) **이동평균** — **매출 이동평균 (5일)**
- "**일별 상품가격 field의 합**”의 (3일) **이동평균** — **매출 이동평균 (3일)**

Data Table Configuration

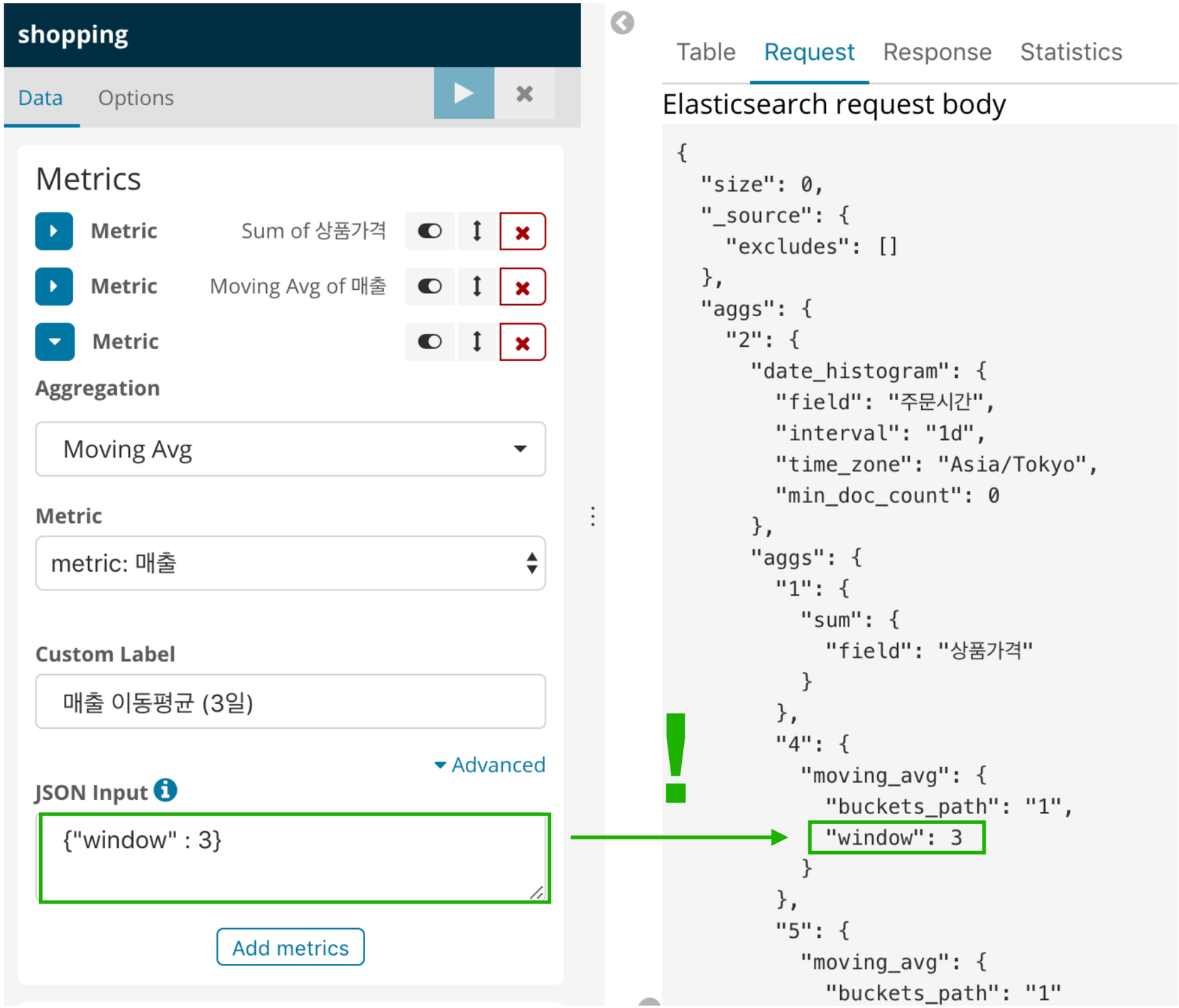
심화



② {“window” : 3} 입력

① Advanced 클릭

뒤에서 이런 작업이 이루어졌다 **심화**



~~빠른 시일 안에 UI 형태로 제공되길...~~

Aggregation 마다 어떤 parameter를 제공하는지 외워야 되는지? **심화**

1. 사용하려는 Aggregation 검색 👑
2. Parameter List에서 Parameter 확인
3. Kibana Visualize 화면 내의 적용하려는 Aggregation 내에서 Advanced 클릭
4. JSON Input에서 변경 *{“parameter name” : “value”}* 형태 입력

Data Table Object 심화

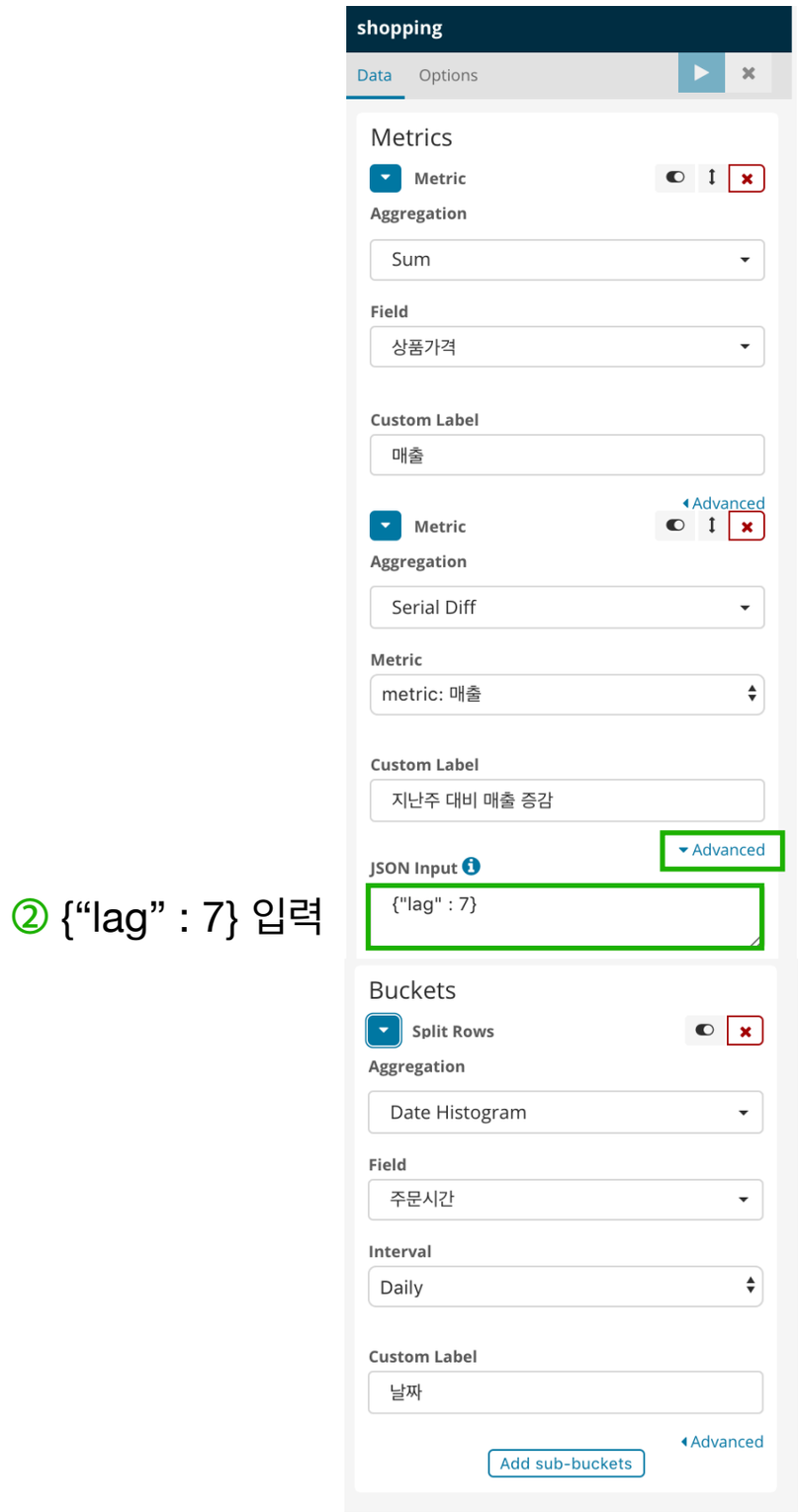
날짜	매출	매출 증감
2018-01-01	1,845,000	-
2018-01-02	1,492,000	-
2018-01-03	1,678,000	-
2018-01-04	1,878,000	-
2018-01-05	1,479,000	-
2018-01-06	1,548,000	-
2018-01-07	1,715,000	-
2018-01-08	1,718,000	-127,000
2018-01-09	1,562,000	70,000
2018-01-10	1,424,000	-254,000

1,562,000 - 1,492,000

조건

- **shopping** index 중에서
- “주문시간” field 기준 **this year** documents의
- “**주문시간**” field로 일별(daily) bucket을 만들고 — **일별**
- “**상품가격**” field의 **합**과 — **매출**
- "**일별 상품가격 field의 합**”의 **증감 (지난주 대비)** — **매출 증감 (지난주 대비)**

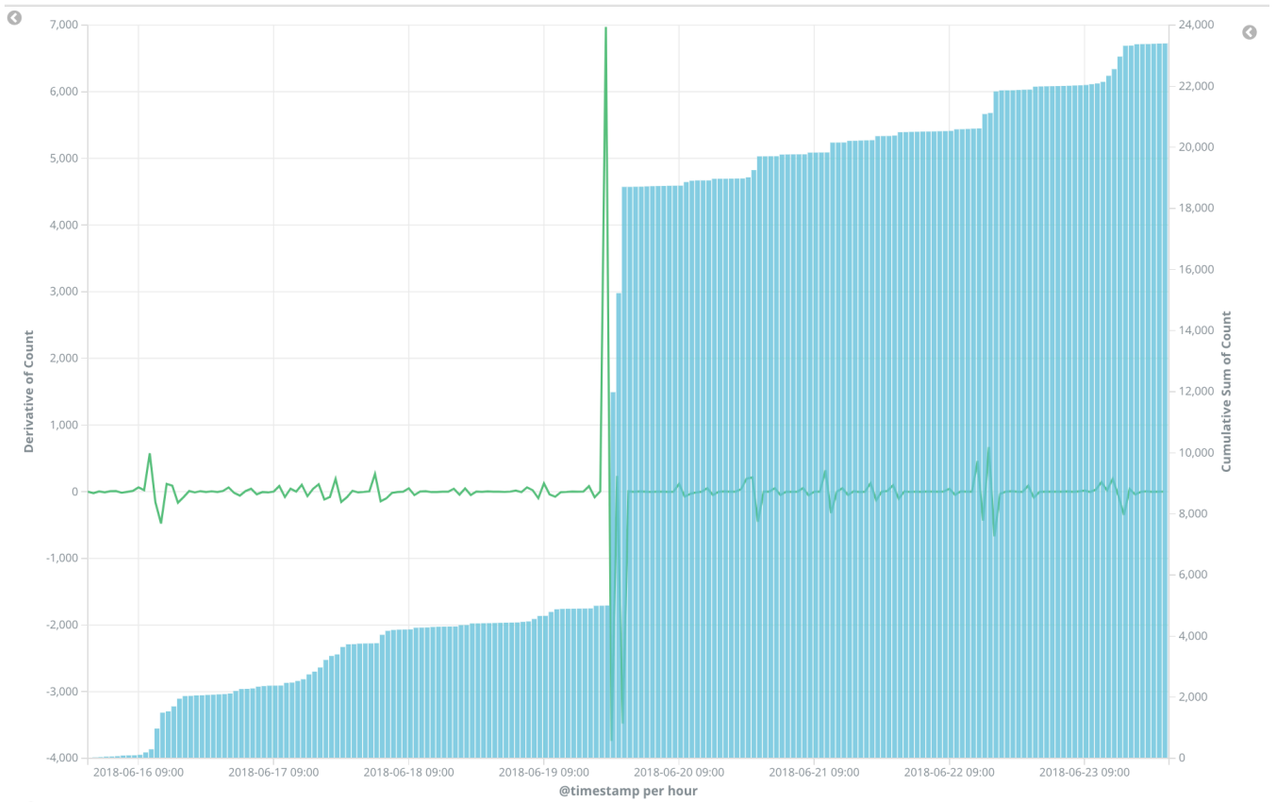
Data Table Configuration 심화



② {“lag” : 7} 입력

① Advanced 클릭

예제 16)



조건

- **nginx-*** index 중에서
- “@timestamp” field 기준 **”2018-06-16 ~ 2018-06-23”** documents를
- **“@timestamp”** field를 기준으로 **시간별** (hourly)로 bucket을 생성한 후의 —— **시간별**
- **documents 개수의 증감** (한 시간 전 대비) 과 —— **접속수 증감 (왼쪽 y축)**
- **documents의 누적 개수** —— **누적 접속수 (오른쪽 y축)**

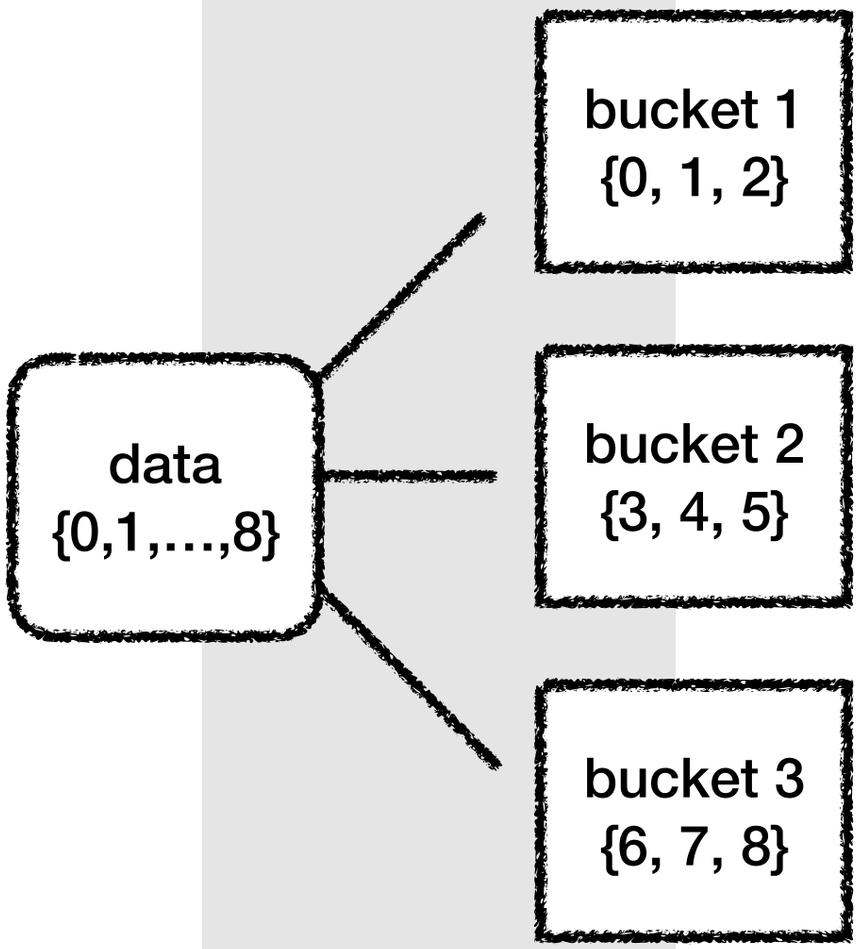
Aggregation - Sibling Pipeline

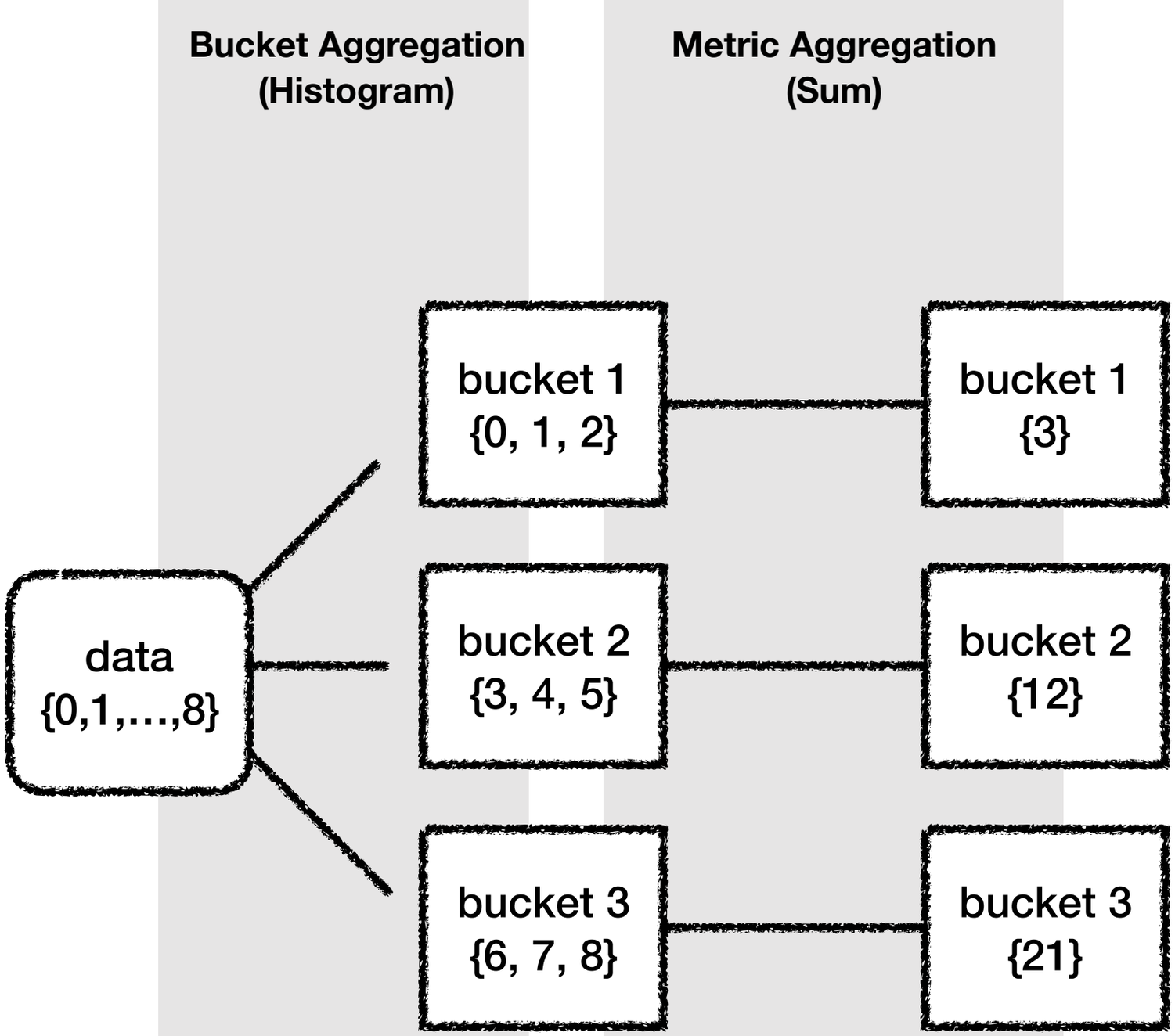
종류	상세
Average Bucket	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Average Agg 적용
Max Bucet	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Max Agg 적용
Min Bucket	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Min Agg 적용
Sum Bucket	Bucket Agg 후, Bucket 별 Metric Agg 하고 난 후, Sum Agg 적용

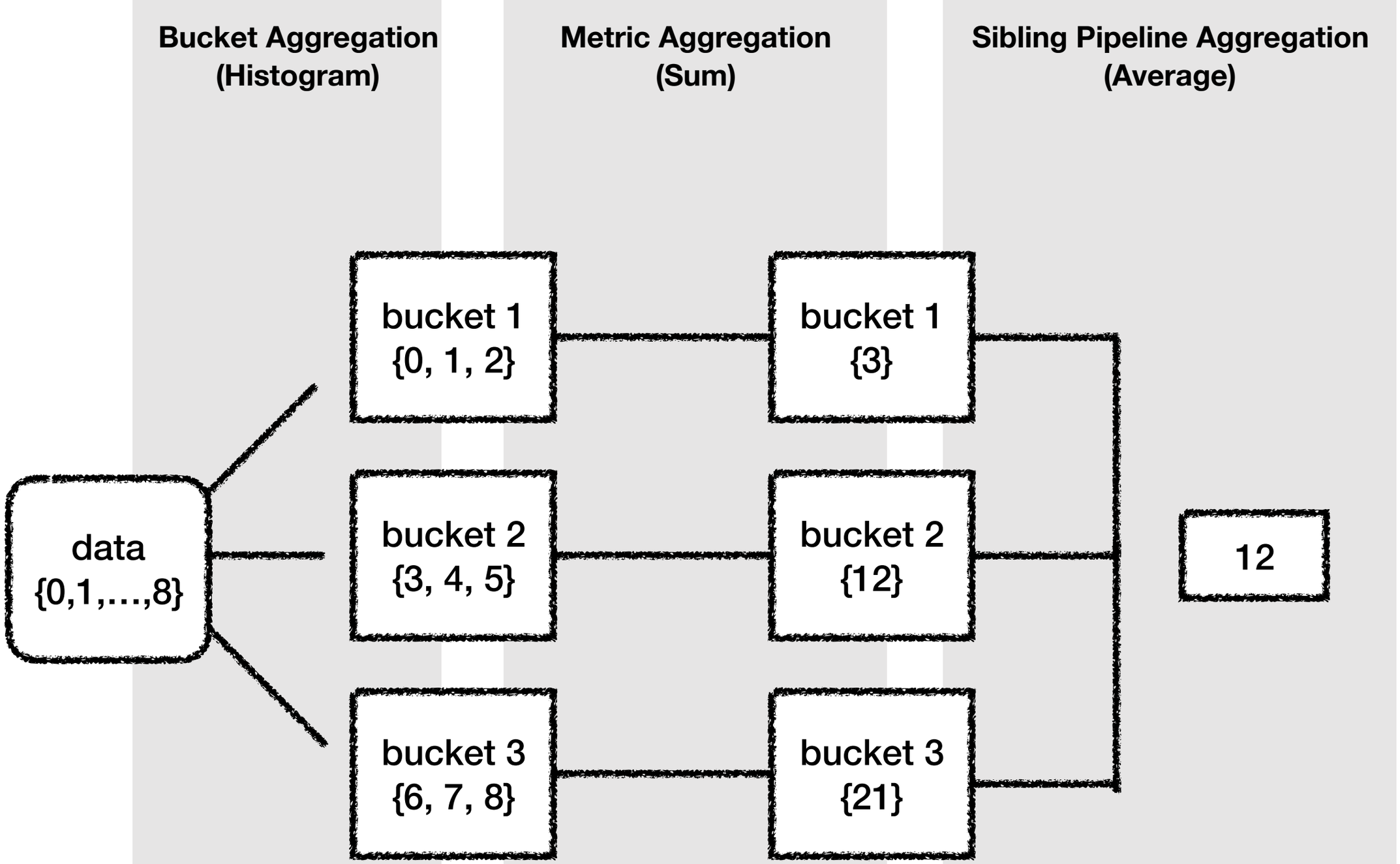
다음과 같은 데이터가 있다고 하자

{“data” : 0}	{“data” : 1}	{“data” : 2}
{“data” : 3}	{“data” : 4}	{“data” : 5}
{“data” : 6}	{“data” : 7}	{“data” : 8}

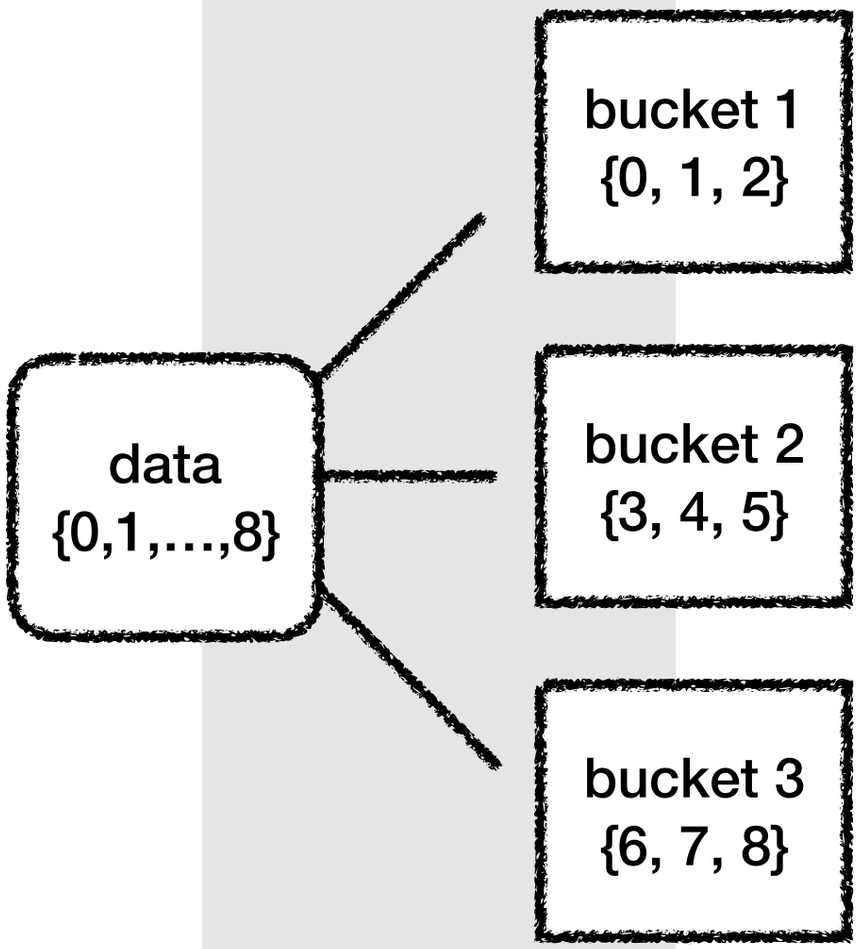
Bucket Aggregation (Histogram)

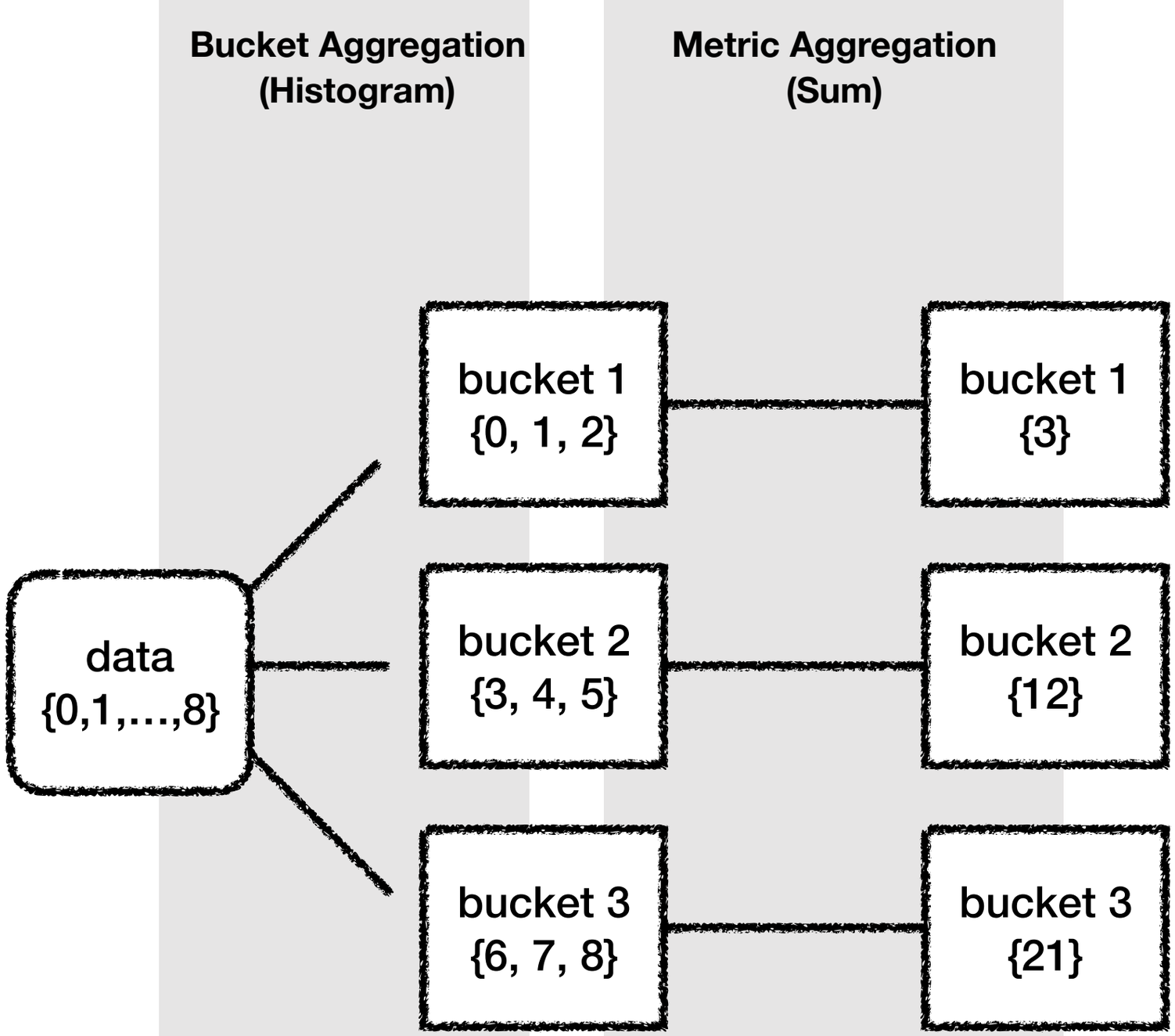


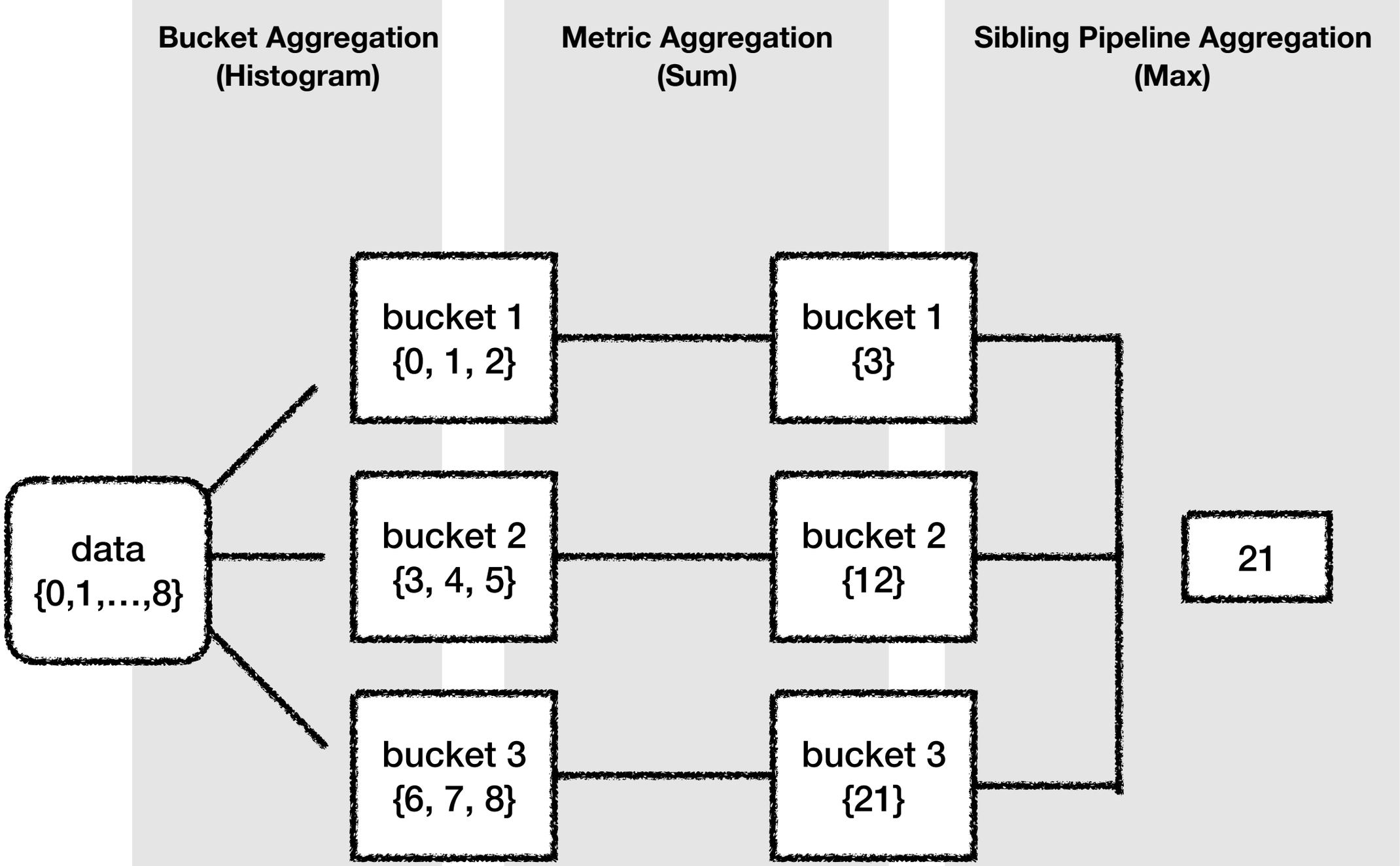

Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}


Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Average)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}
12

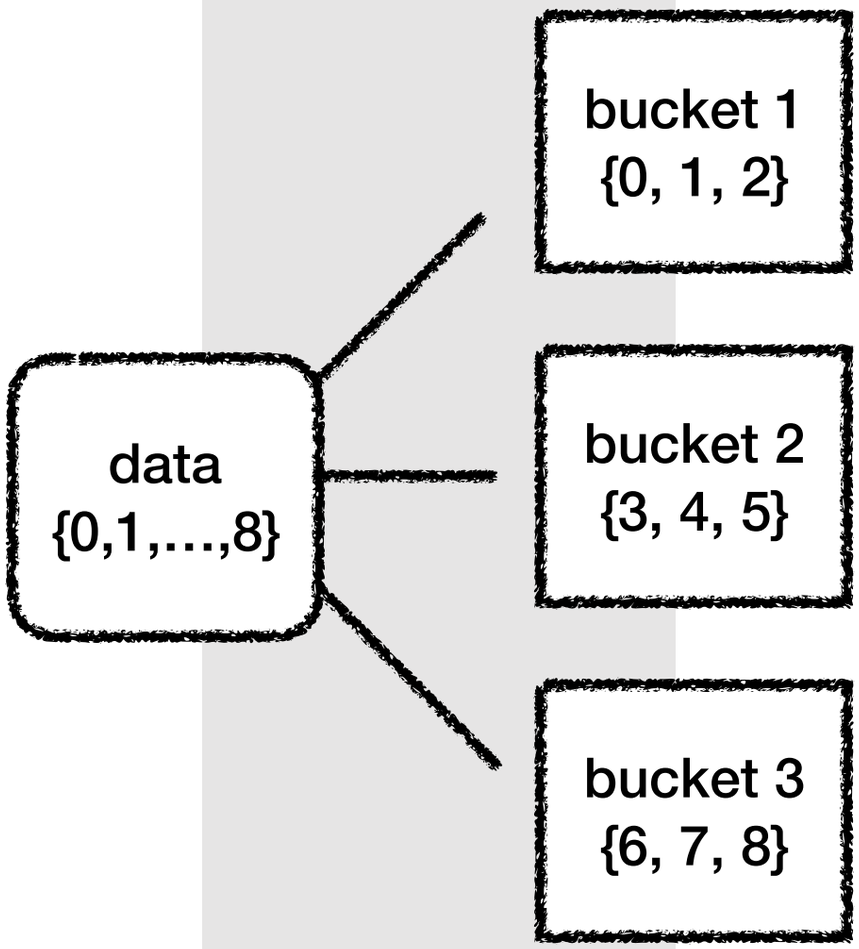
Bucket Aggregation (Histogram)

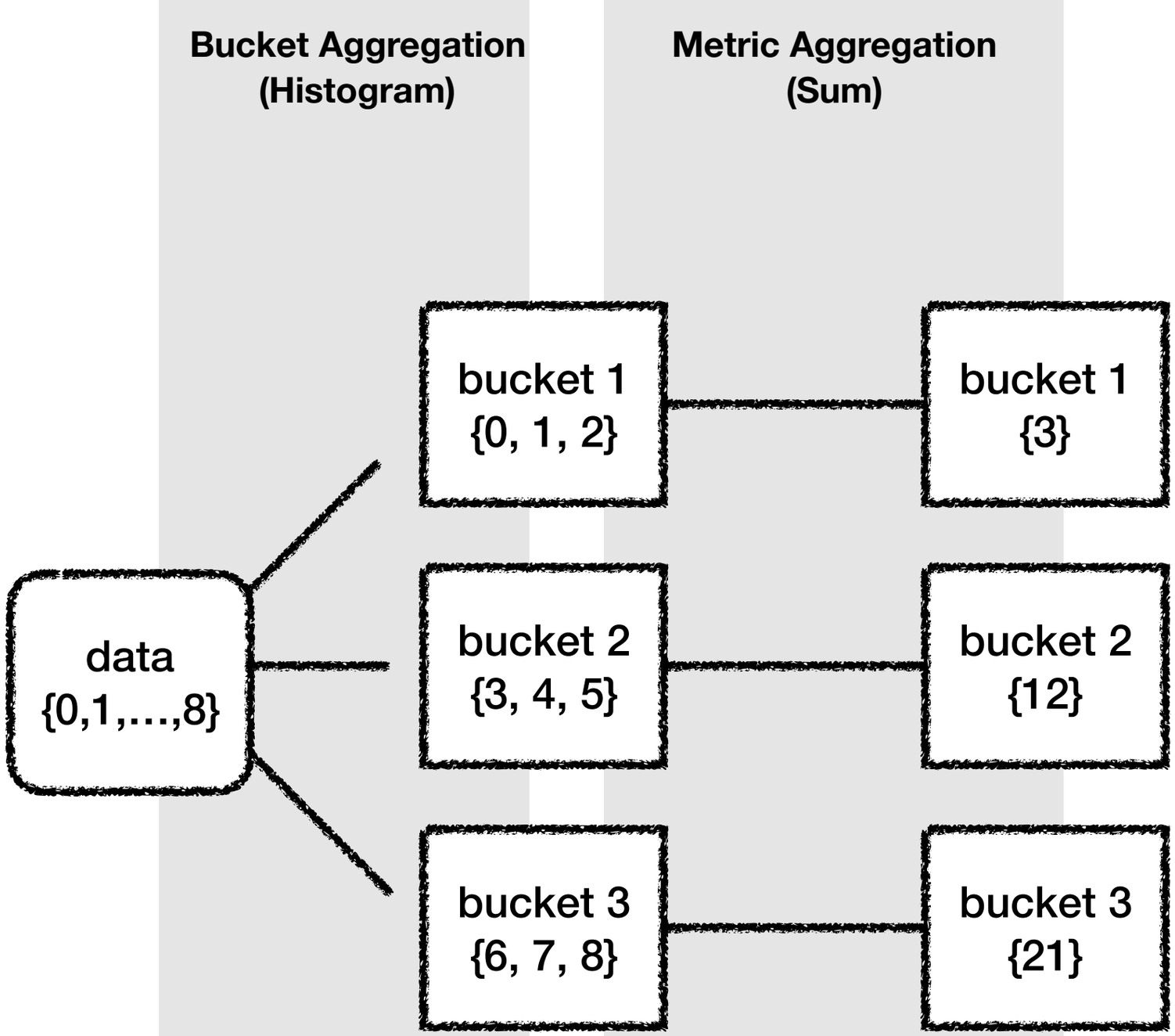


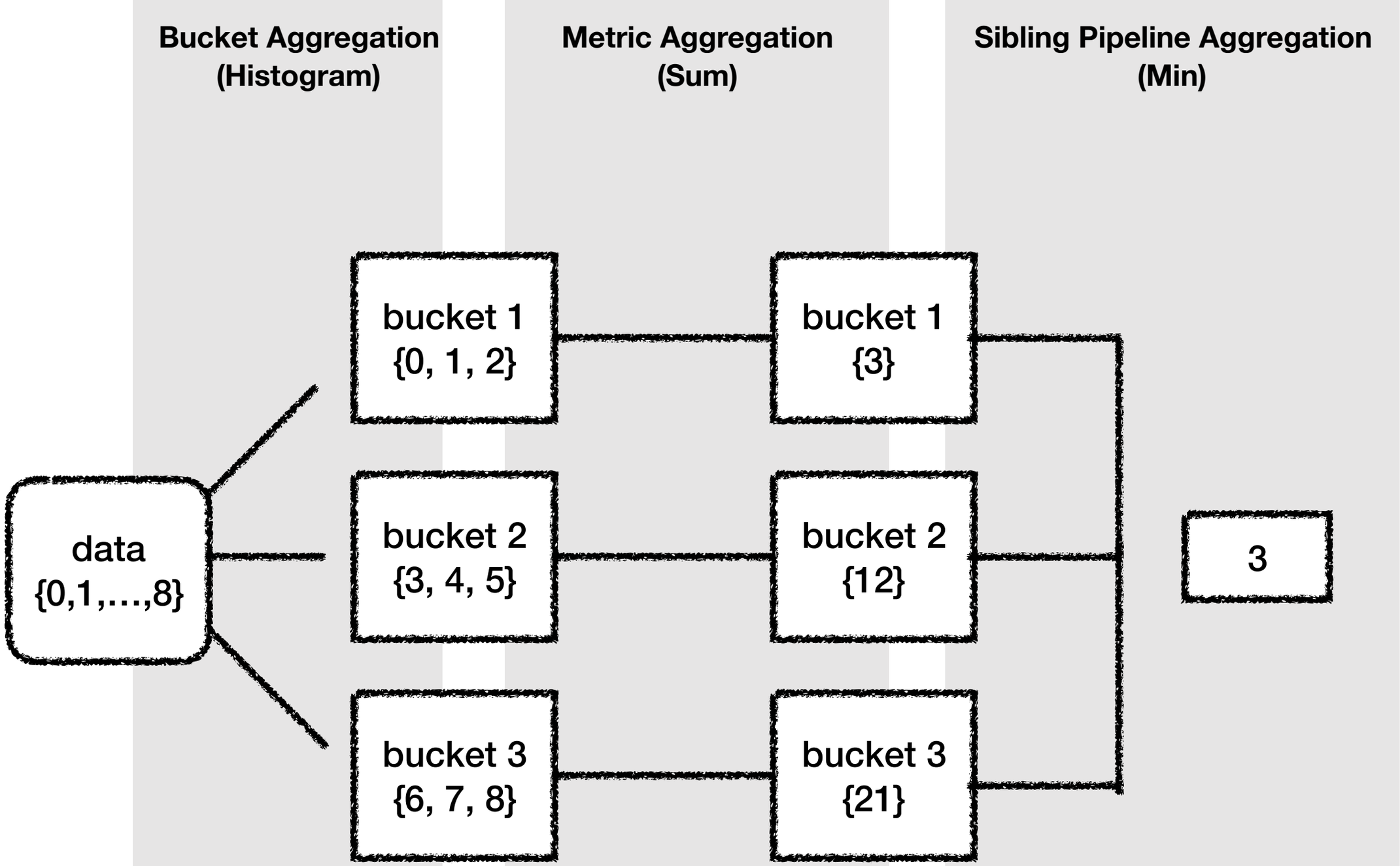

Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}


Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Max)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}
21

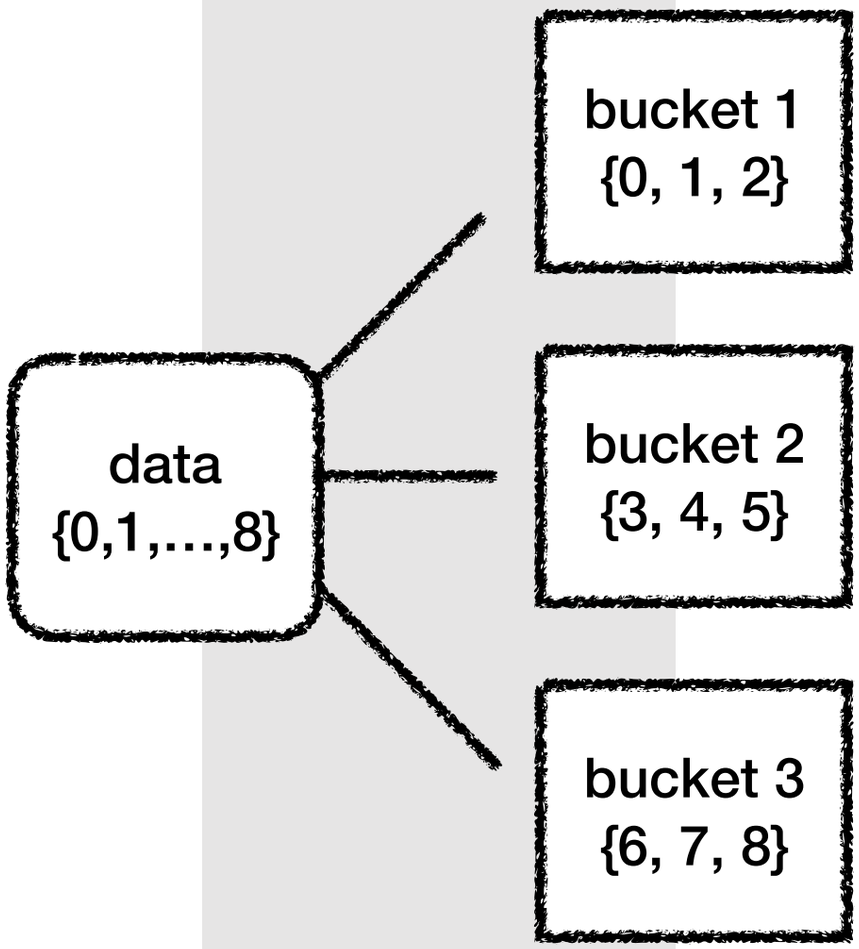
Bucket Aggregation (Histogram)

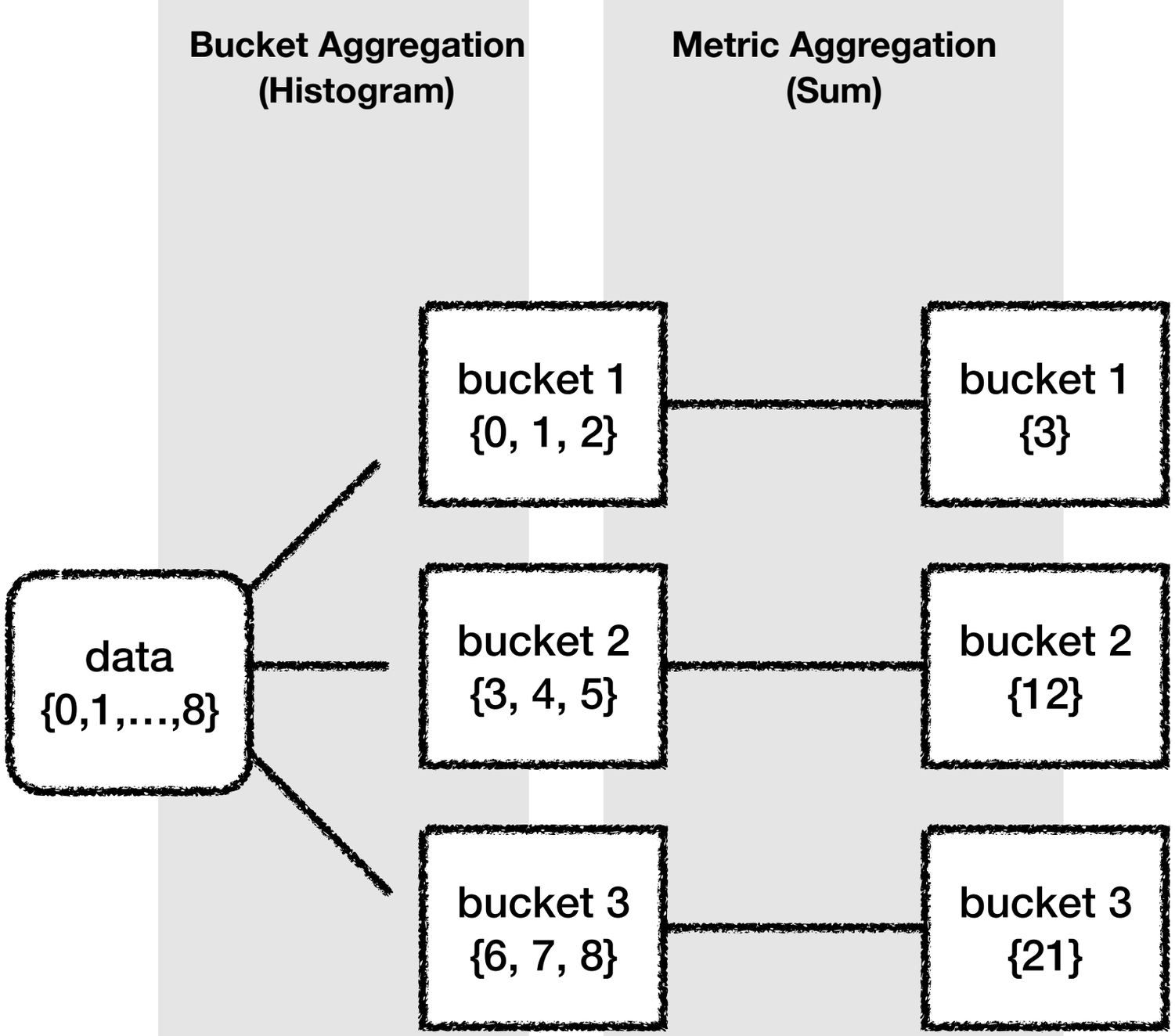


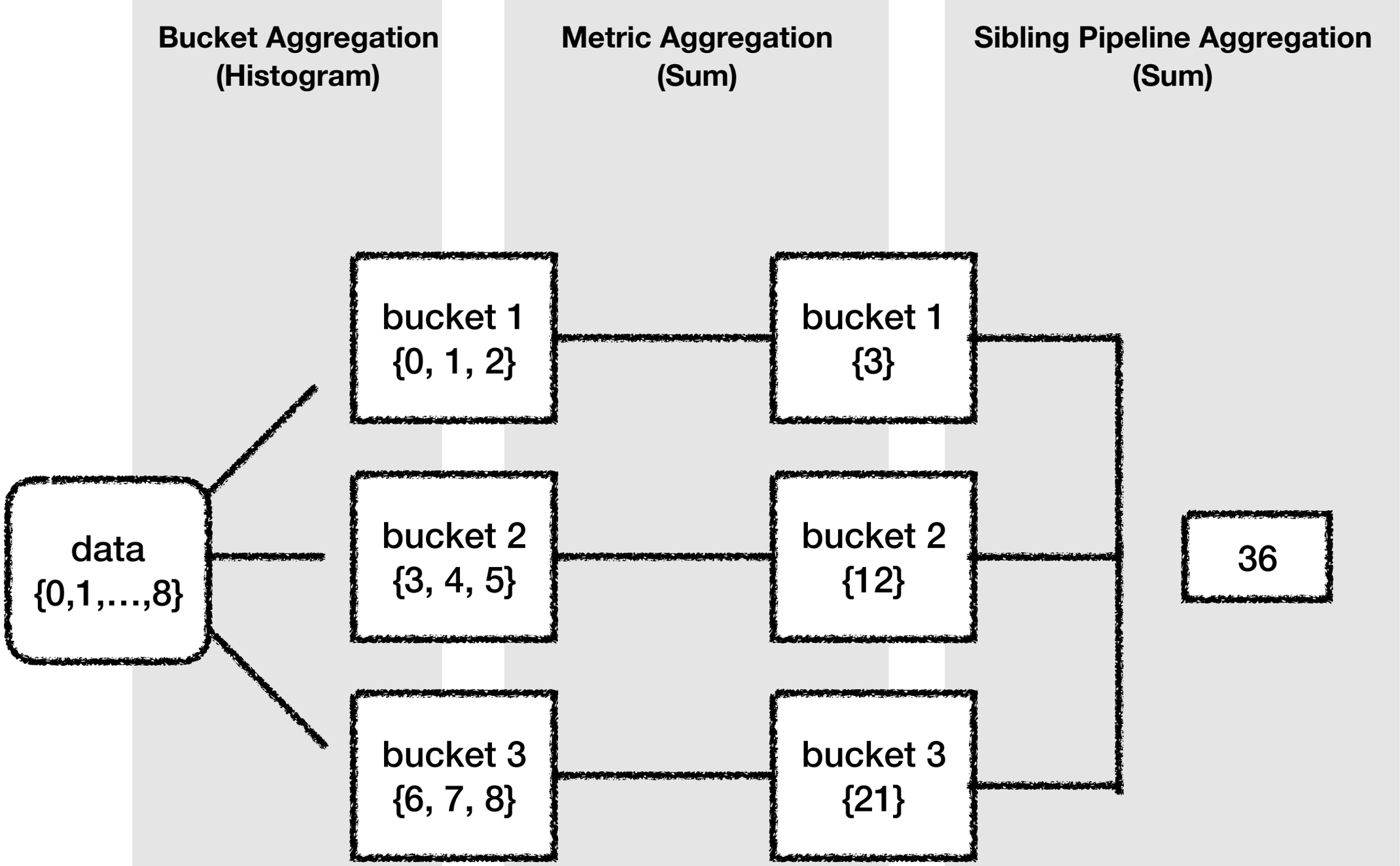

Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}


Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Min)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}
3

Bucket Aggregation (Histogram)




Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}


Bucket Aggregation
(Histogram)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Sum)
data
{0,1,…,8}
bucket 1
{0, 1, 2}
bucket 2
{3, 4, 5}
bucket 3
{6, 7, 8}
bucket 1
{3}
bucket 2
{12}
bucket 3
{21}
36

실제 Kibana에서 **Sibling** Pipeline Aggregation을 어떻게 사용하는지 보자

Metric Object

63,179,012
Windows 10 - Overall Average of Sum of nginx.access.body_sent.bytes

42,940,913.667
Mac OS X - Overall Average of Sum of nginx.access.body_sent.bytes

조건

- **nginx-*** index 중에서
- “@timestamp” field 기준 **“2018-06-11 ~ 2018-08-12”** documents의
- **documents가 가장 많았던 “nginx.access.user_agent.os” field 1개**에 대해서 각각 — **os별**
  - **documents가 가장 많았던 “nginx.access.geoip.city_name” field 3개**의 — **(접속이 많았던 도시 3개의) 데이터 합의**
  - **“nginx.access.body_sent.bytes”** field값을 **합**한 후의
  - **평균** — **평균**

Metric Object

63,179,012

Windows 10 - Overall Average of Sum of nginx.access.body_s

940,913.667

Overall Average of Sum of nginx.access.body_sent.bytes

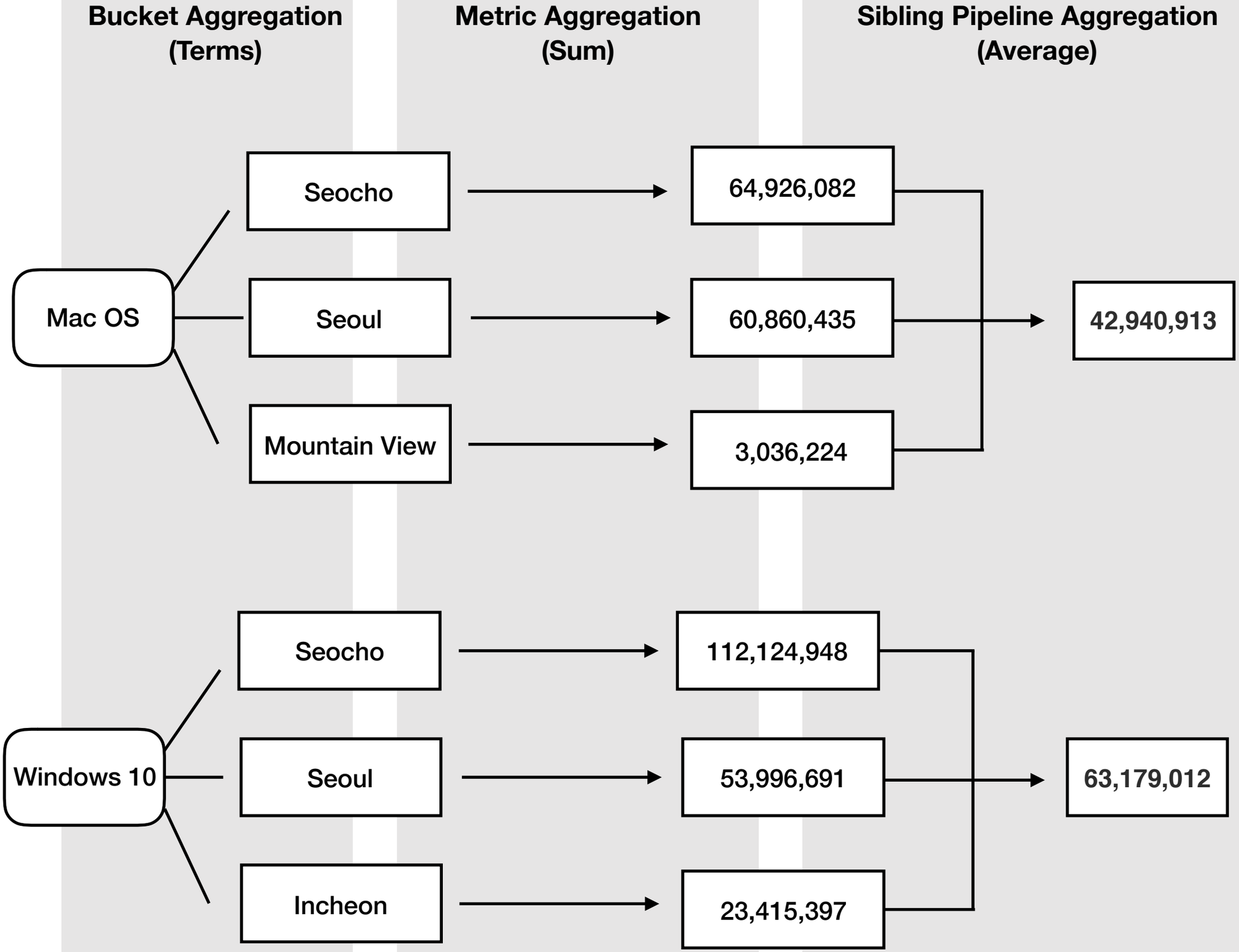
조건

- **nginx-*** index 중에서
- “@timestamp” field 기준 **“2018-06-11 ~ 2018-08-12”** documents의
- **documents가 가장 많았던** “**nginx.access.user_** eld **1개**에 대해서 각각
  - **documents가 가장 많았던** “**nginx.access.g** **ame**” field **3개**의
  - “**nginx.access.body_sent.bytes**” field값을
  - **평균**

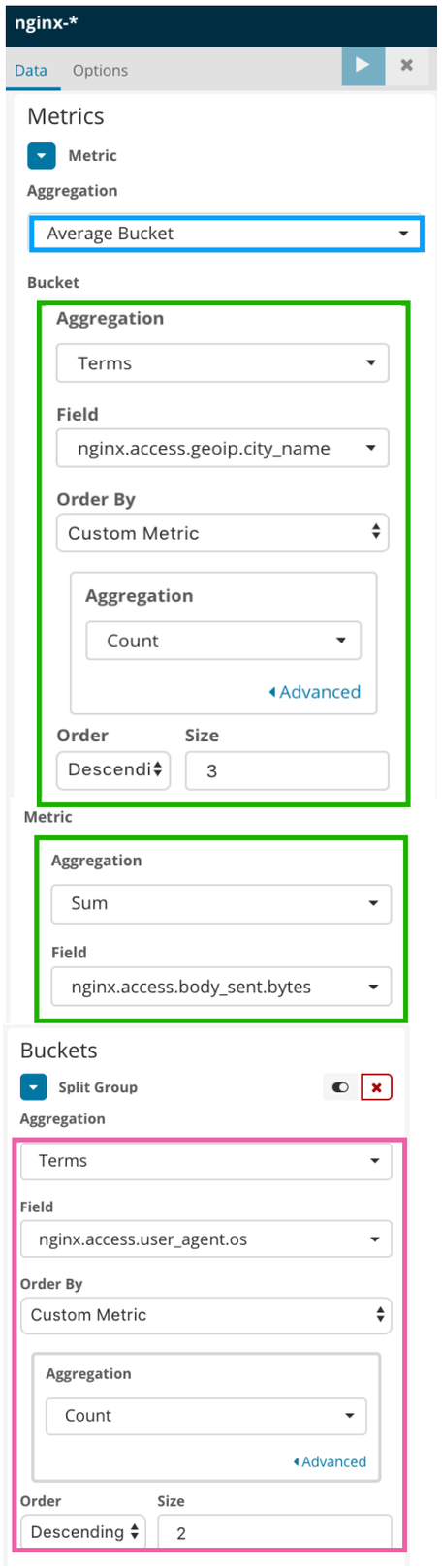
— **os별**

┐ **(접속이 많았던 도시 3개의) 데이터 합의**

— **평균**


Bucket Aggregation
(Terms)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Average)
Mac OS
Seocho
Seoul
Mountain View
64,926,082
60,860,435
3,036,224
42,940,913
Windows 10
Seocho
Seoul
Incheon
112,124,948
53,996,691
23,415,397
63,179,012

Metric Configuration



② Average Bucket Aggregation 선택

③

(Average Bucket Aggregation에서 필요한) Bucket을 생성하기 위한 Aggregation 선택

④

바로 위의 ③에서 생성한 각 bucket에 적용할 metric aggregation 선택

① Terms Aggregation 설정

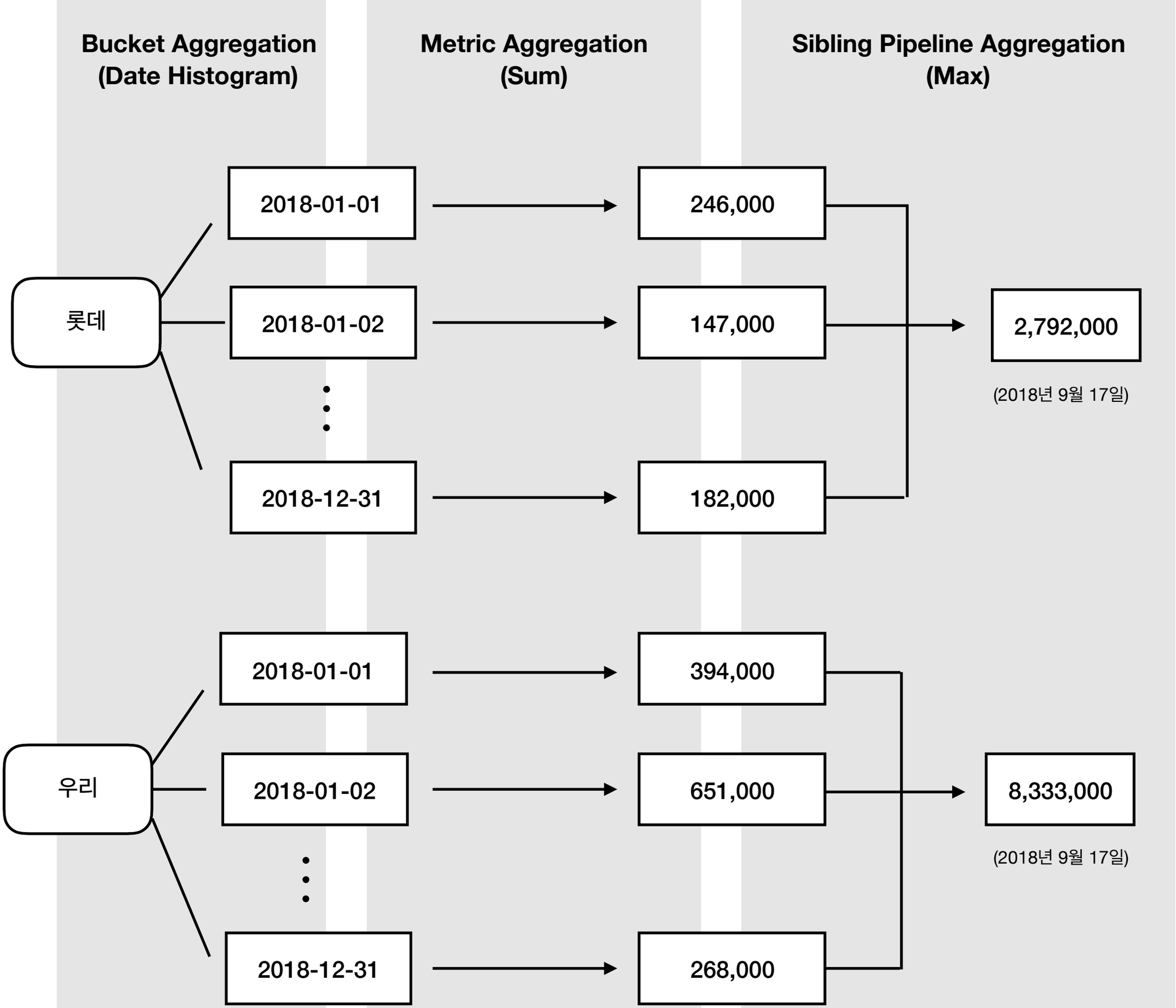
Metric Object

2,792,000
롯데 - Overall Max of Sum of 상품가격

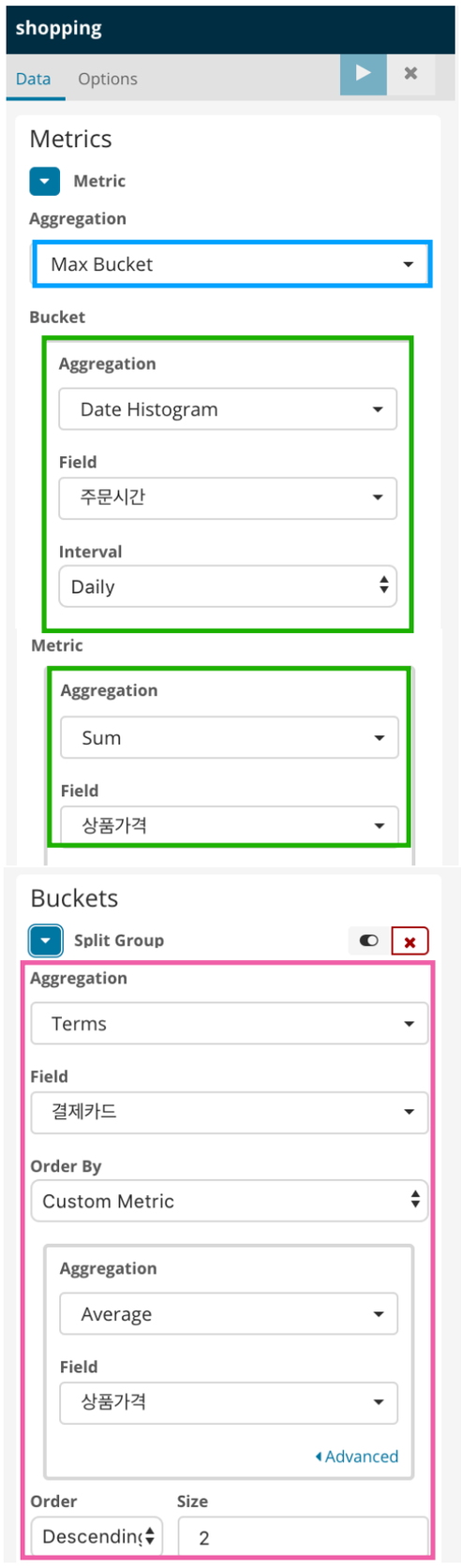
8,333,000
우리 - Overall Max of Sum of 상품가격

조건

- **shopping** index 중에서
- “ 주문시간” field 기준 **“this year”** documents의
- “**상품가격**” field의 **평균이 가장 컸던** “**결제카드**” field **2개**에 대해서 각각 — **카드별**
  - “**주문시간**” field를 기준으로 **일별** (daily) — **일별**
  - “**상품가격**” field의 **합**을 구한 후의 — **매출의**
  - **최대값** — **최대값**


Bucket Aggregation
(Date Histogram)
Metric Aggregation
(Sum)
Sibling Pipeline Aggregation
(Max)
롯데
2018-01-01
2018-01-02
2018-12-31
246,000
147,000
182,000
2,792,000
(2018년 9월 17일)
우리
2018-01-01
2018-01-02
2018-12-31
394,000
651,000
268,000
8,333,000
(2018년 9월 17일)

Metric Configuration



② Max Bucket Aggregation 선택

③

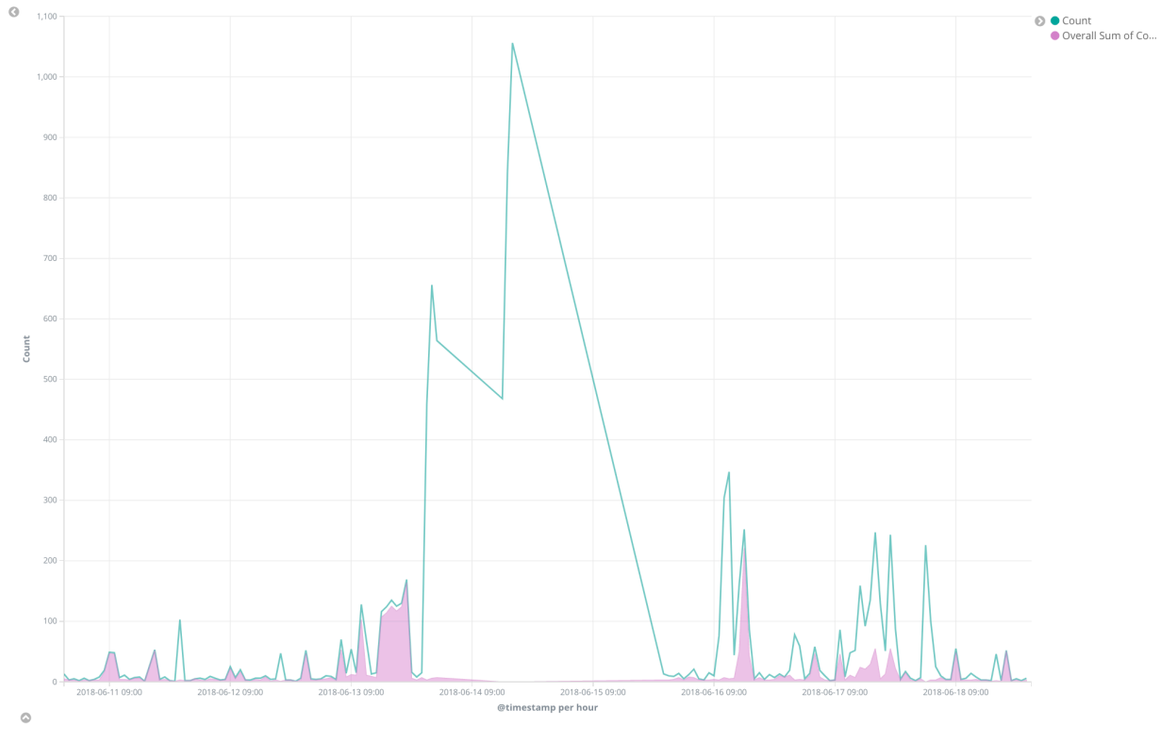
(Average Bucket Aggregation에서 필요한) Bucket을 생성하기 위한 Aggregation 선택

④

바로 위의 ③에서 생성한 각 bucket에 적용할 metric aggregation 선택

① Terms Aggregation 설정

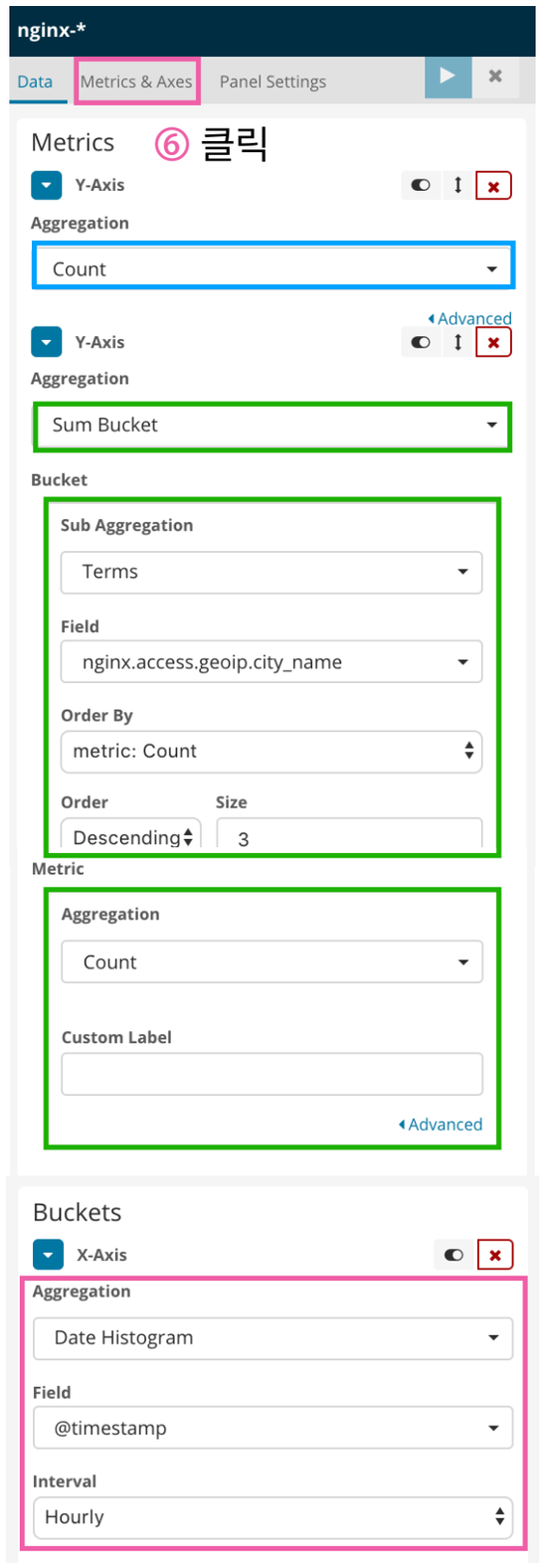
Line Chart Object



조건

- **nginx-*** index 중에서
- “@timestamp” field 기준 **“2018-06-11 ~ 2018-06-18”** documents를
- “**@timestamp**” field를 기준으로 **시간별** (hourly)로 bucket을 생성한 후의 —— **시간별**
  1) documents 개수와
  2)
     - **documents 개수가 가장 많았던 “nginx.access.geoip.city_name”** field **3개**의 ┐ **(접속이 많은 도시 3개의) 접속수의**
     - **document 개수**의 ┘
     - **합** —— **합**

Line Chart Configuration1



② Count Aggregation 선택 (전체 접속수 표시)

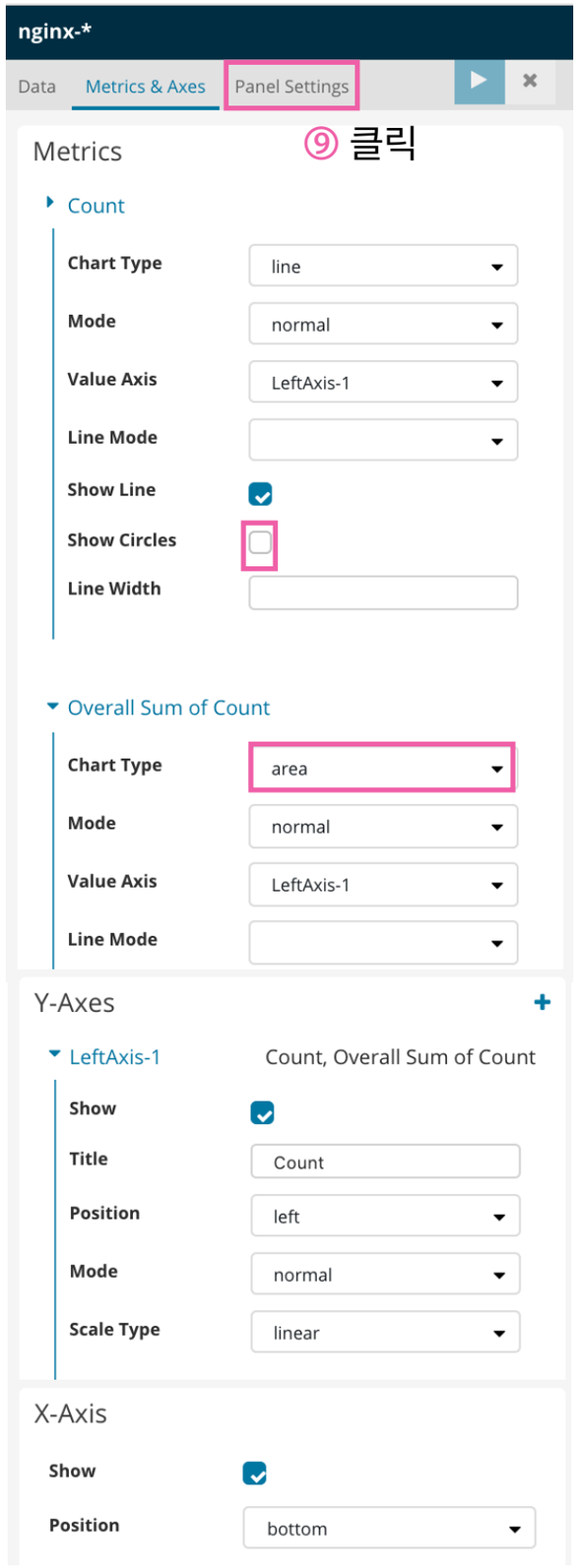
③ Sum Bucket Aggregation 선택

④
(Sum Bucket Aggregation에서 필요한)
Bucket을 생성하기 위한 Aggregation 설정

⑤
바로 위의 ④에서 생성한 각 bucket에 적용할 metric aggregation 선택

① Date Histogram Aggregation 설정

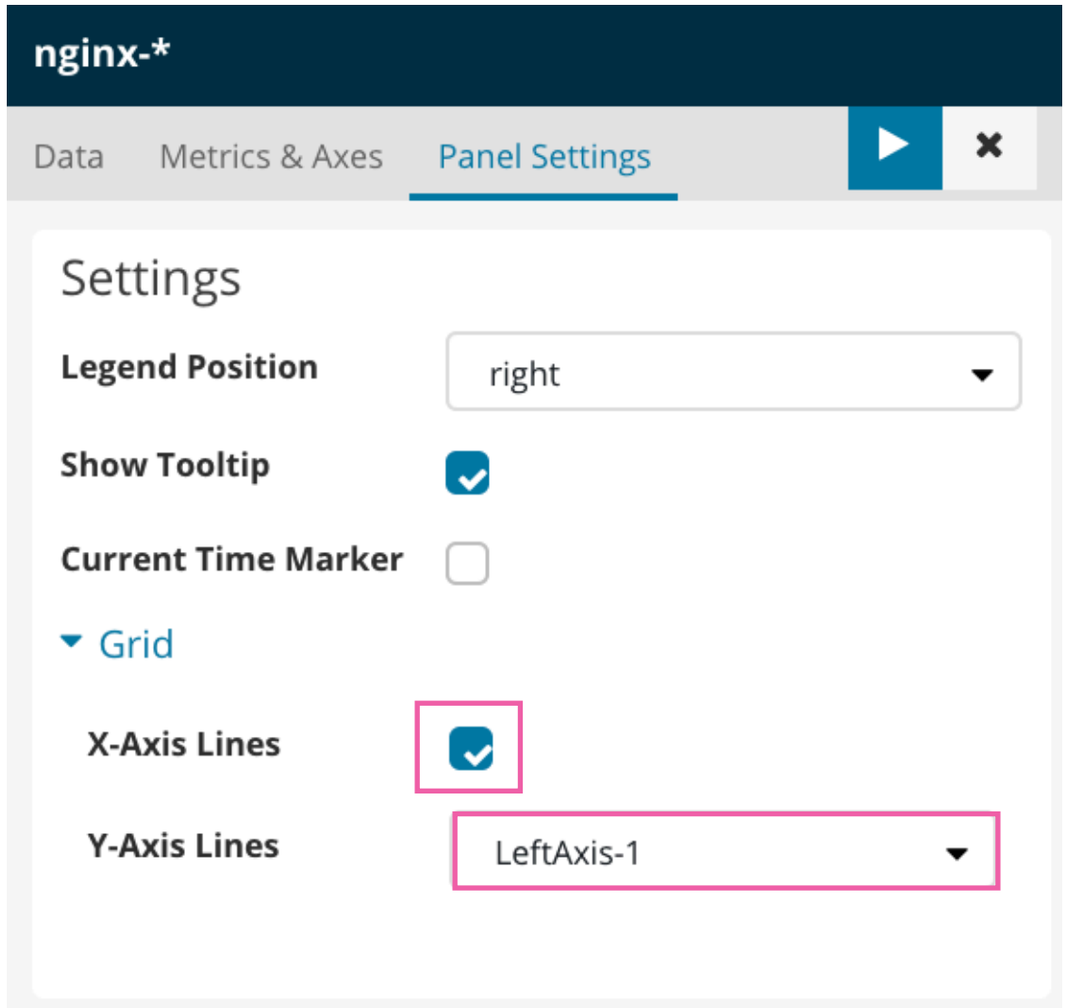
Line Chart Configuration 2



⑦ 해제

⑧ area 선택

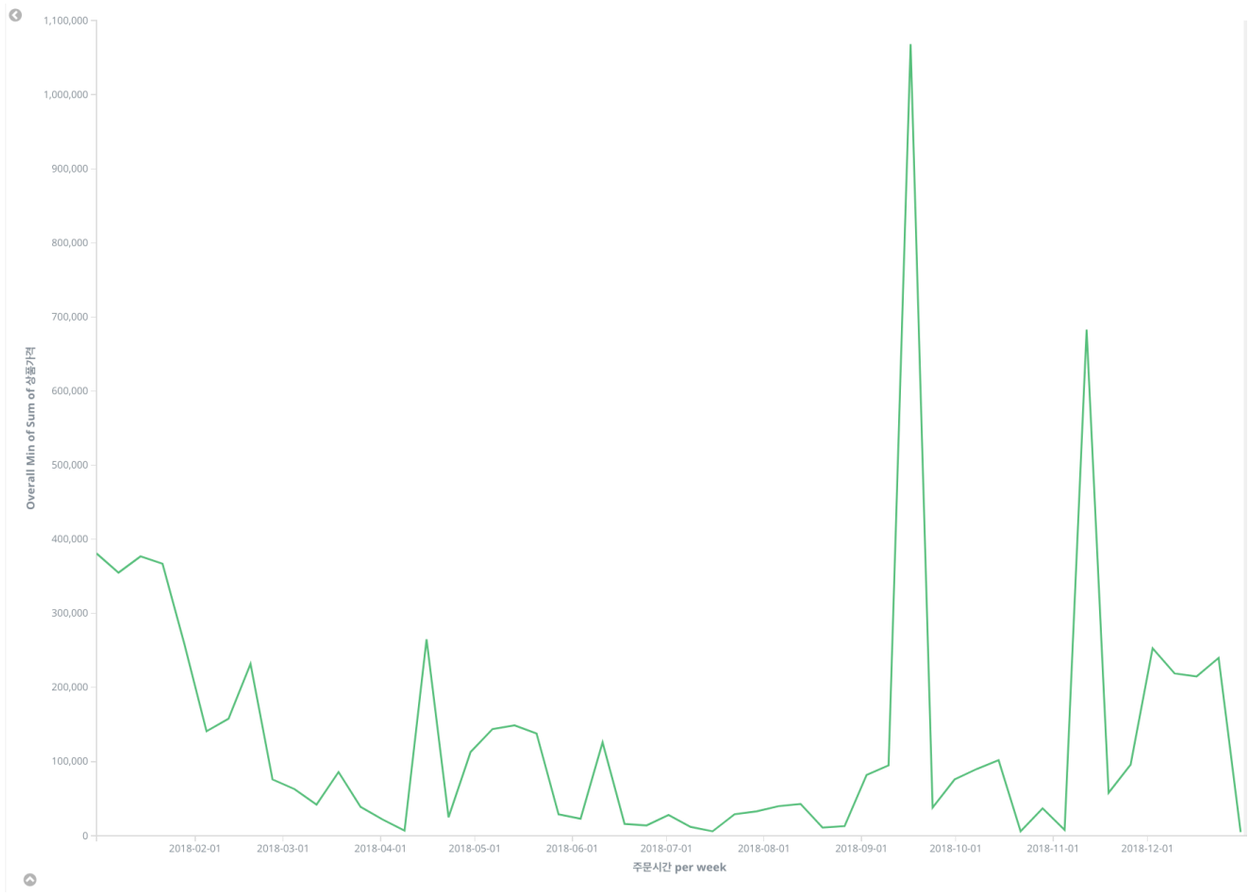
Line Chart Configuration 3



⑩ X-Axis Lines 선택

⑪ LeftAxis-1 선택

예제 17)



조건

- **shopping** index 중에서
- “주문시간” field 기준 **this year** documents를
- “**주문시간**” field를 기준으로 **주별** (weekly)로 bucket을 생성한 후의
  - “**가나다**” 순서상 **상위 5개** “**상품분류**” field에 대해서
  - “**상품가격**” field의 **합**의
  - **최소값**

주별

(5개) 상품별 매출의

최소값

→ **주별로 (5개 상품중) 가장 매출이 적은 상품의 매출 추적**

질문 및 Feedback은

gshock94@gmail.com로 주세요