filebeat / kafka / nifi / Elasticsearch / Kibana 데이터 파이프라인 구축

응용통계학과 신보현

INDEX

ch 1 주제 및 아키텍쳐

ch 2 개발 환경

ch 3 결과

ch 4 심화

ch 5 느낀점

Chapter 1

주제 및 아키텍처

주제

kafka을 중간다리 역할로 데이터 파이프 라인을 구축해보자

1. 데이터를 수집하고
2. 수집한 데이터를 저장하며
3. 저장한 데이터를 시각화하는

일련의 과정을 직접 구현해보자

주제

HOW?

1. 데이터를 수집 -> kafka
2. 수집한 데이터를 저장 -> Elasticsearch
3. 저장한 데이터를 시각화 -> kibana

어떤 데이터를 수집할까

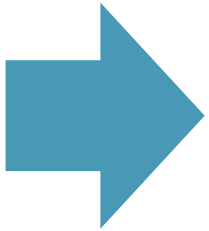
- 지속적으로 발생하는 데이터라면 좋다
- 인위적으로 만드는 것보다는 의미있는 데이터로 하고 싶었다
- kafka에서 발생하는 로그는 필요할 때 모니터링하면 좋으므로 카프카의 로그 데이터를 수집하기로 결정하였다.
- 로그 데이터는 filebeat로 수집하였다.

프로젝트 아키텍쳐

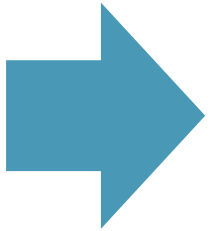
수집된 로그 데이터는 카프카의 log-topic에 저장됨

filebeat에서 카프카의 로그 데이터 수집 (producer 역할)

nifi가 log-topic에 있는 로그 데이터를 consume함 (consumer 역할)



nifi에 있는 로그 데이터를 elasticsearch에 가져옴



elasticsearch에 있는 로그 데이터를 kibana로 시각화함

Chapter 2

개발 환경

개발 환경

1. AWS에 kafka 클러스터, zookeeper 서버를 구축

- kafka는 총 3대의 브로커로 구성
- zookeeper는 짝수면 안되고, 1대면 의미가 없으므로 3대로 구성
- 이때, kafka, zookeeper를 각기 다른 6개의 EC2 서버에 구축함
- 추후에 nifi와 zookeeper을 동일한 서버에서 구동하면 문제가 발생할 수 있기 때문에 아예 분리하여 구축함

개발 환경

1. AWS에 kafka 클러스터, zookeeper 서버를 구축

<카프카 클러스터>

test-broker01,

test-broker02,

test-broker03

으로 지정

카프카 버전: 2.12

<주키퍼 서버>

test-zk01,

test-zk02,

test-zk03

으로 지정

주키퍼 버전: 3.5.9

개발 환경

카프카 토픽 (log-topic) 구성 및 주키퍼 설정

- acks=1

-> 현 프로젝트에서 데이터 유실이 큰 문제가 되지 않으므로 acks=all로 지정하지 않음

- partition: 3
- replication factor: 2
- 주키퍼 최상위 노드가 아니라 지노드를 구분해서 사용(test-zk01:2181,test-zk02:2181,test-zk03:2181/test)

-> 주키퍼 앙살블 세트를 공용으로 사용하기 위해서

개발 환경

2. test-broker01/03에 filebeat을 설치

- 카프카의 로그를 받아오기 위해 경량 데이터 수집기인 filebeat을 사용
- 모든 브로커에 설치하지 않고 두대에만 설치함

개발 환경

2. test-broker01에 filebeat을 설치

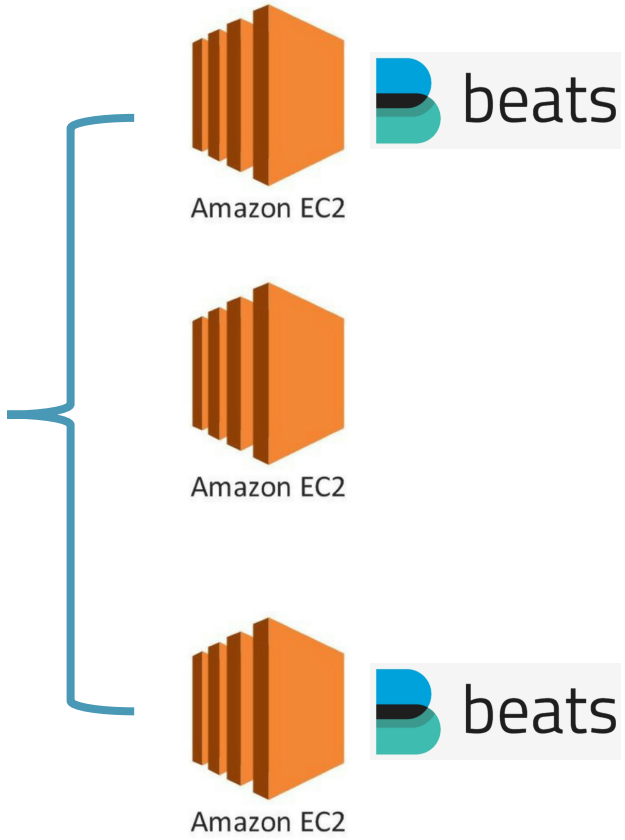
<카프카 클러스터>

test-broker01,

test-broker02,

test-broker03

으로 지정



<주키퍼 서버>

test-zk01,

test-zk02,

test-zk03

으로 지정

카프카 버전: 2.12

주키퍼 버전: 3.5.9

개발 환경

3. 카프카 브로커에 nifi을 설치

- nifi와 주키퍼를 동일한 서버에 설치하면 문제가 발생할 수 있음
- 따라서 카프카 브로커에 nifi을 설치함
- 또한 최신 nifi 버전은 주키퍼 3.5.x 이상과 호환이 잘 됨

개발 환경

3. 카프카 브로커에 nifi를 설치

<카프카 클러스터>

test-broker01,

test-broker02,

test-broker03

으로 지정

<주키퍼 서버>

test-zk01,

test-zk02,

test-zk03

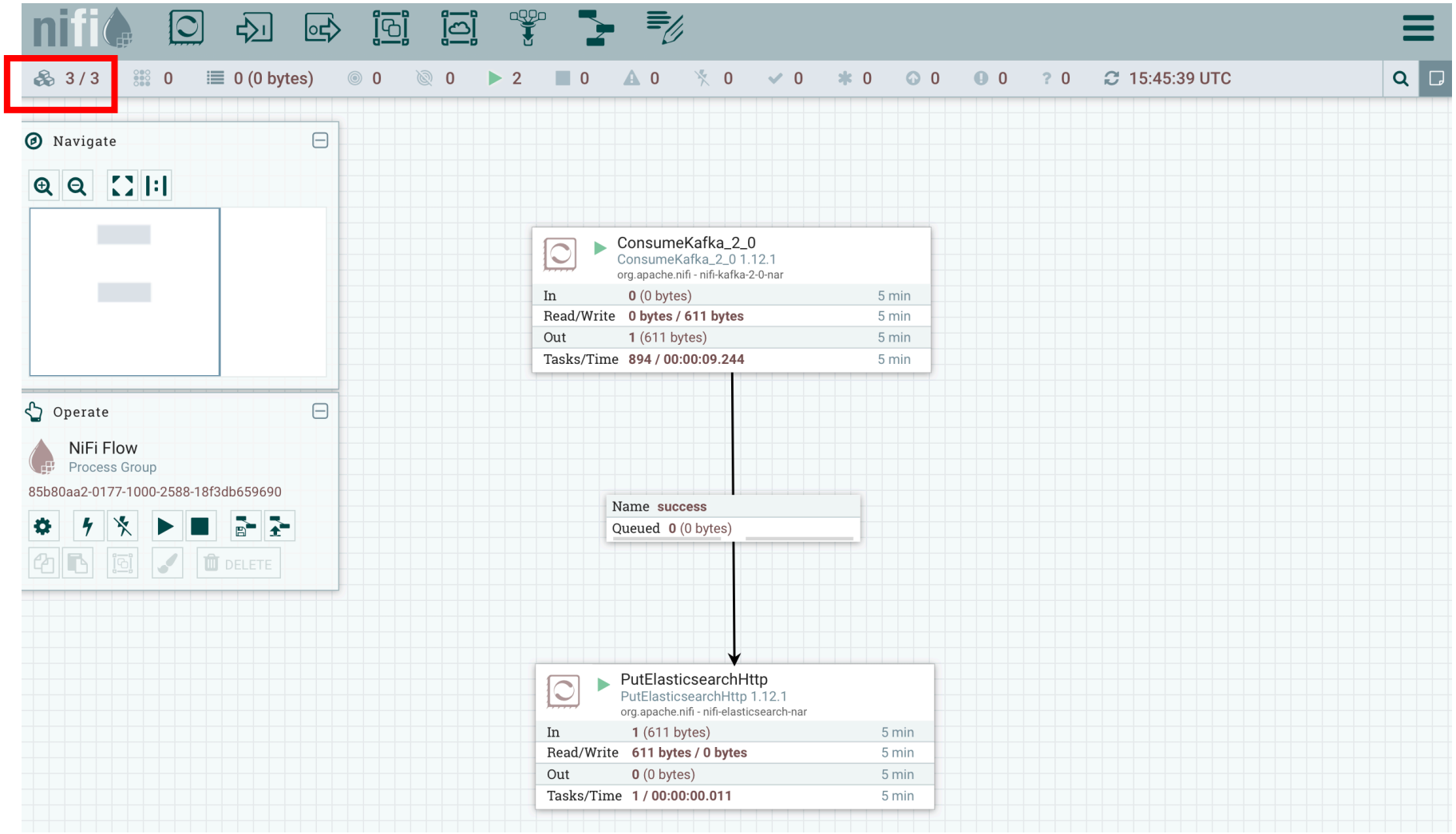
으로 지정

카프카 버전: 2.12 **nifi 버전: 1.12**

주키퍼 버전: 3.5.9

개발 환경

3. 카프카 브로커에 nifi을 설치: web에 띄운 모습



개발 환경

4. test-broker02에 elasticsearch, kibana 설치

- 동일한 브로커에 설치함
- root에서 실행이 안되므로 일반 계정에서 설치 후 실행

개발 환경

4. test-broker02에 elasticsearch, kibana 설치

<카프카 클러스터>

test-broker01,

test-broker02,

test-broker03

으로 지정

<주키퍼 서버>

test-zk01,

test-zk02,

test-zk03

으로 지정

카프카 버전: 2.12 **nifi 버전: 1.12** **주키퍼 버전: 3.5.9**

개발 환경

최종 Jps 결과

<카프카 클러스터>

test-broker01,

test-broker02,

test-broker03

으로 지정

```
13137 Jps
8893 Elasticsearch
```

```
5362 RunNiFi
5379 NiFi
4824 Kafka
9294 Jps
```

<주키퍼 서버>

test-zk01,

test-zk02,

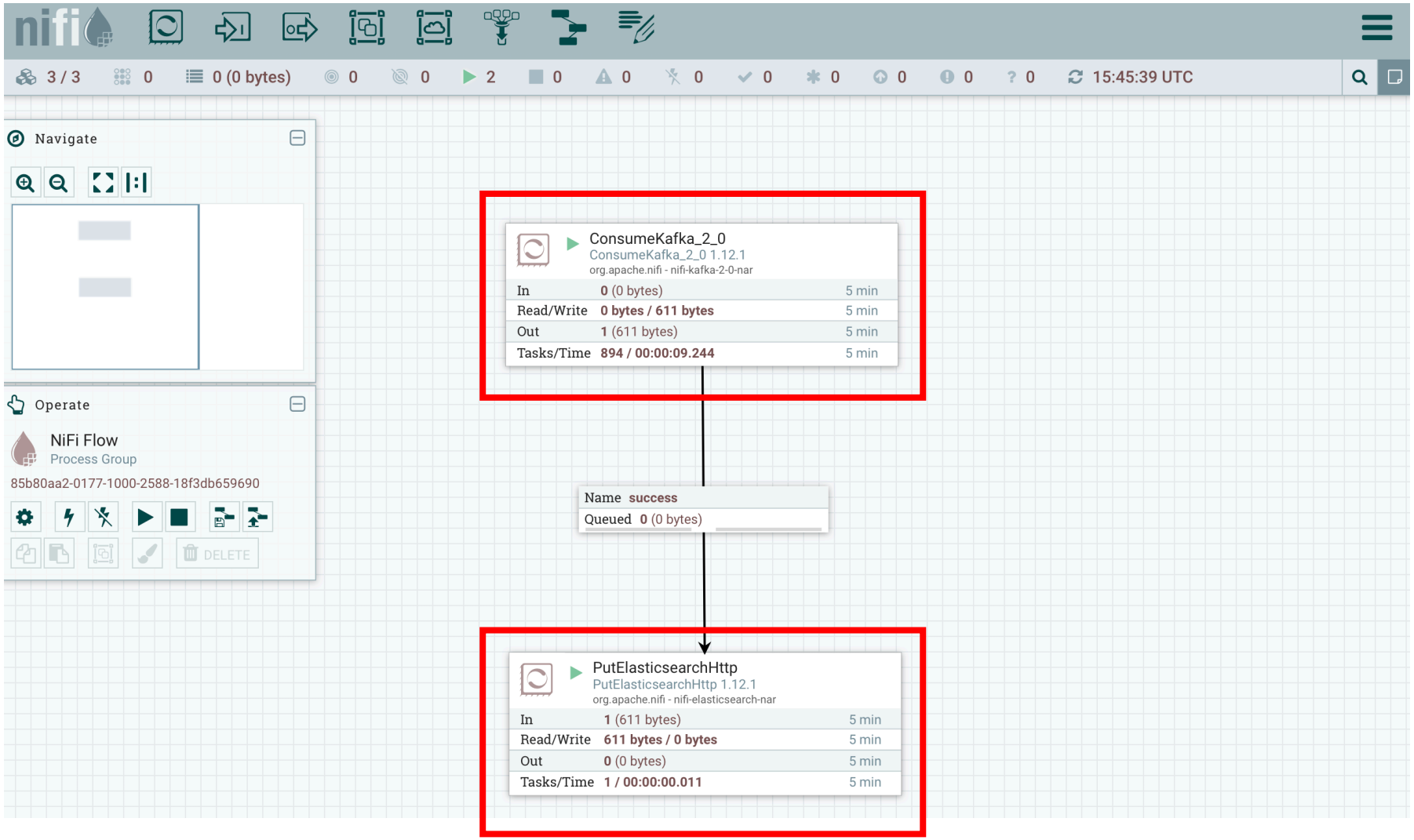
test-zk03

으로 지정

```
3282 QuorumPeerMain
5228 Jps
```

개발 환경

nifi을 통해서 kafka와 elasticsearch을 연결한 모습



개발 환경

elasticsearch가 성공적으로 띄워진 경우

curl –XGET을 날려보면 오른쪽과 같은 응답을 받을 수 있다.

```
{
  "name" : "test-broker02",
  "cluster_name" : "test-ES",
  "cluster_uuid" : "L02vK1_sQJe4Byi9QEdClQ",
  "version" : {
    "number" : "7.6.2",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "ef48eb35cf30adf4db14086e8aabd07ef6fb113f",
    "build_date" : "2020-03-26T06:34:37.794943Z",
    "build_snapshot" : false,
    "lucene_version" : "8.4.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

개발 환경

kibana가 성공적으로 띄워진 경우

아래와 같은 로그를 확인할 수 있다.

```
  log   [01:44:01.030] [info][listening] Server running at http://0.0.0.0:5601
  log   [01:44:01.282] [info][      ][Kibana][http] http server running at http:/
/0.0.0.0:5601
```

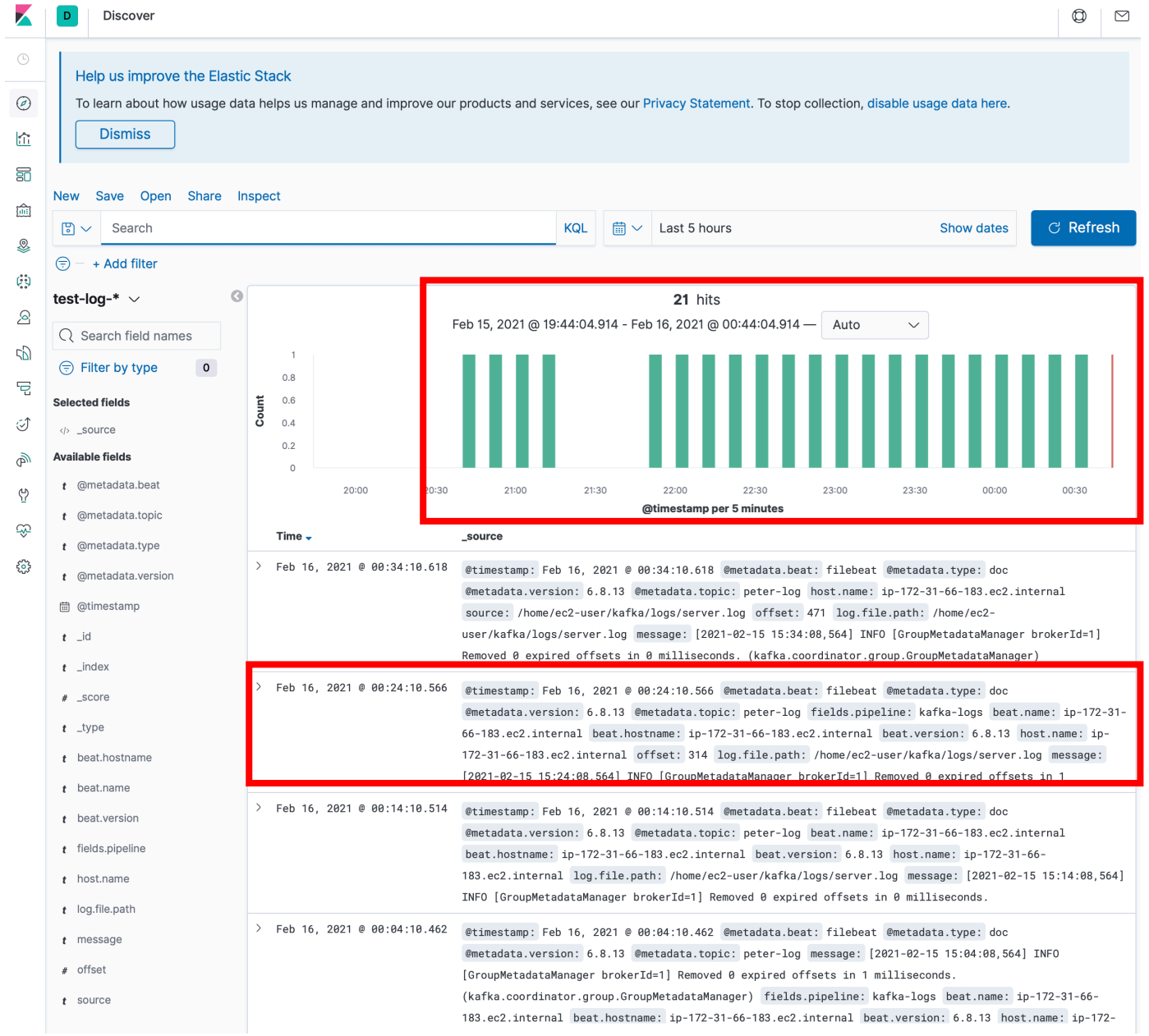
Chapter 3

결과

결과

kibana 시각화 결과

- 시간대별로 로그 개수를 확인할 수 있음
- 각 로그의 내용을 일목요연하게 확인할 수 있음



Chapter 4

심화

심화

데이터 파이프라인을 더 깊게 구축해보자

1. test-log 토픽에는 test-broker01/03의 카프카 로그들이 쌓이고 있다.
2. 이 로그들을 카프카의 각 브로커별로 구분할 필요가 있다.
3. test-log 토픽에 쌓인 두 브로커의 카프카 로그들을 각기 다른 카프카 토픽으로 옮겨보자

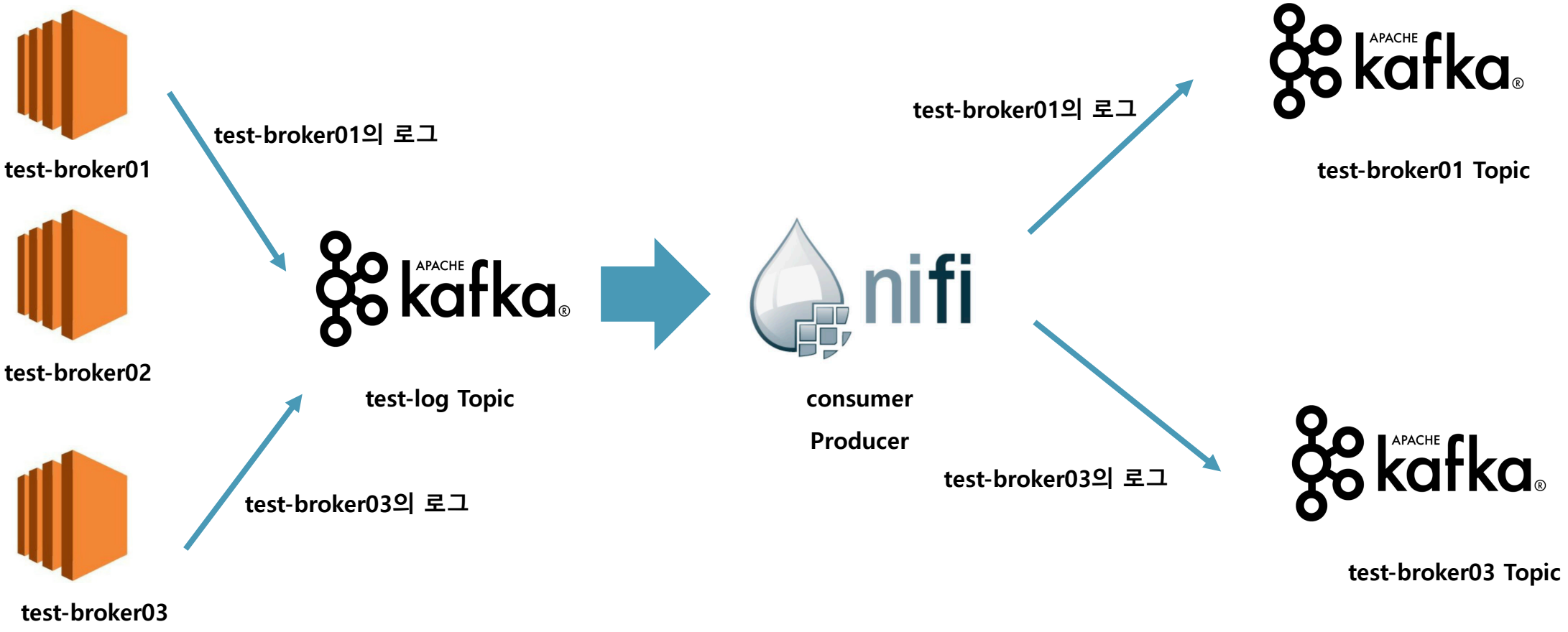
심화

처음부터 두 브로커의 로그를 각기 다른 토픽에 쌓으면 안될까?

1. 메세지를 전송하는 퍼블리셔가 전송해야하는 토픽의 수가 많아질수록 부담이 될 수 있음
2. 따라서 한 토픽에 저장된 메세지를 다시 다른 토픽으로 옮기는 경우가 종종 있다고 함

심화

데이터 흐름도


test-broker01
test-broker02
test-broker03
test-broker01의 로그
test-broker03의 로그
APACHE kafka
test-log Topic
nifi
consumer
Producer
test-broker01의 로그
test-broker03의 로그
APACHE kafka
test-broker01 Topic
APACHE kafka
test-broker03 Topic

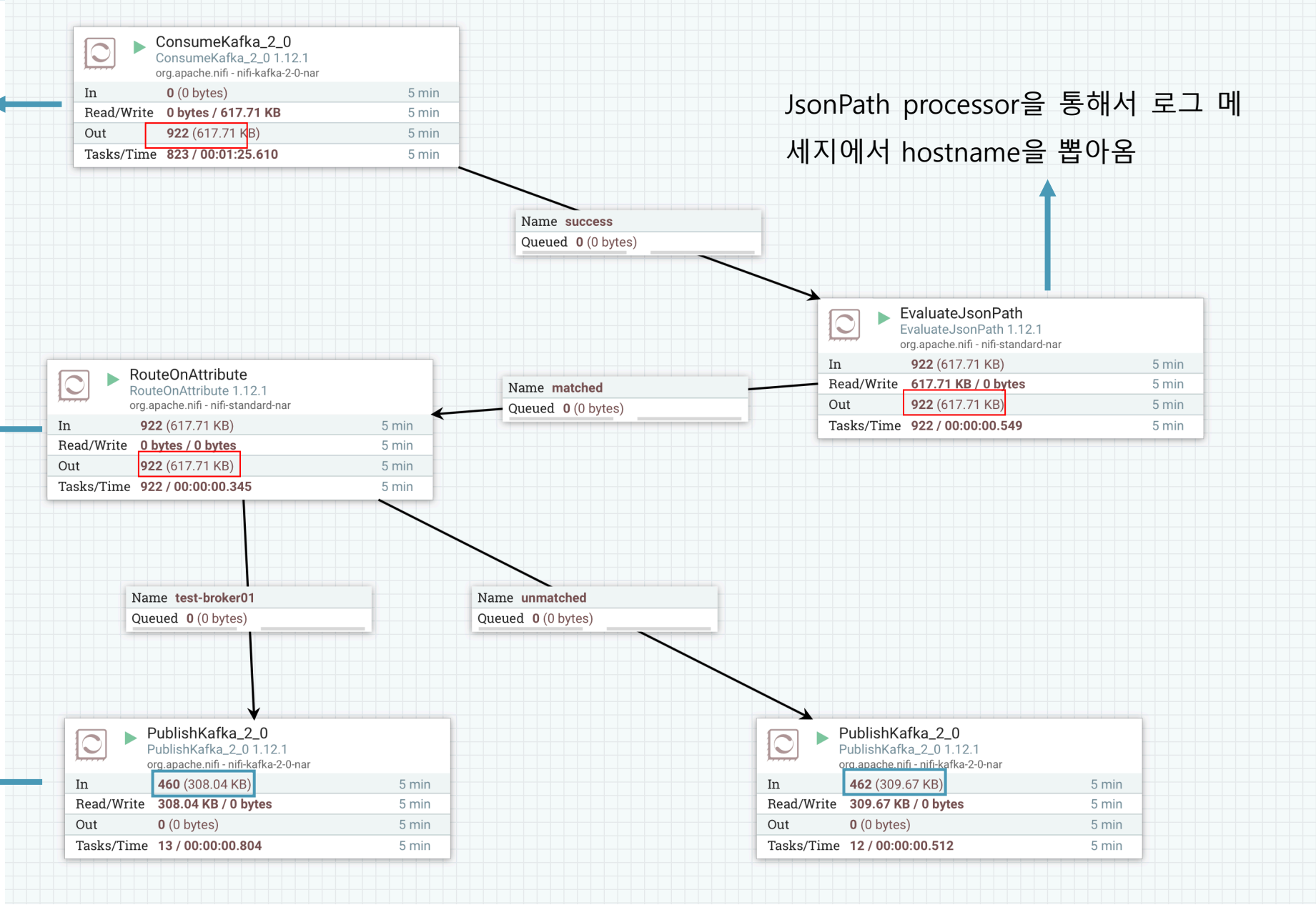
nifi로 연결한 데이터 흐름도

test-log 토픽으로 총 922개의 메세지가 유입됨. 여기에는 test-broker01/03의 로그 메세지가 섞여 있음

JsonPath processor을 통해서 로그 메세지에서 hostname을 뽑아옴

Route Processor을 통해서 hosname이 test-broker01인지를 구분해냄(T/F 값으로 반환)

460개의 메세지는 test-broker01, 462개의 메세지는 test-broker03에서 발생된 것이며 이들의 합은 초기 922개의 메세지와 동일함. 즉 데이터 유실이 발생하지 않음



심화

test-broker01 토픽에 쌓인 메세지 확인


```
{"@timestamp":"2021-02-16T04:22:14.767Z","@metadata":{"beat":"filebeat","type":"
doc","version":"6.8.13","topic":"peter-log"},"fields":{"pipeline":"kafka-logs"},
"beat":{"name":"ip-172-31-66-183.ec2.internal","hostname":"ip-172-31-66-183.ec2.
internal","version":"6.8.13"},"host":{"name":"ip-172-31-66-183.ec2.internal"},"s
ource":"/home/ec2-user/kafka/logs/server.log","offset":103577,"log":{"file":{"pa
th":"/home/ec2-user/kafka/logs/server.log"}},"message":"[2021-02-16 04:22:09,792
] INFO [GroupMetadataManager brokerId=1] Group console-consumer-11402 transition
ed to Dead in generation 2 (kafka.coordinator.group.GroupMetadataManager)"}
{"@timestamp":"2021-02-16T04:22:14.767Z","@metadata":{"beat":"filebeat","type":"
doc","version":"6.8.13","topic":"peter-log"},"host":{"name":"ip-172-31-66-183.ec
2.internal"},"log":{"file":{"path":"/home/ec2-user/kafka/logs/server.log"}},"mes
sage":"[2021-02-16 04:22:09,796] INFO [GroupMetadataManager brokerId=1] Removed
0 expired offsets in 8 milliseconds. (kafka.coordinator.group.GroupMetadataManag
er)","source":"/home/ec2-user/kafka/logs/server.log","offset":103755,"fields":{"
pipeline":"kafka-logs"},"beat":{"name":"ip-172-31-66-183.ec2.internal","hostname
":"ip-172-31-66-183.ec2.internal","version":"6.8.13"}}
```

심화

test-broker03 토픽에 쌓인 메세지 확인


```
{"name":"ip-172-31-78-33.ec2.internal"},"source":"/home/ec2-user/kafka/logs/serv
er.log","offset":111021,"log":{"file":{"path":"/home/ec2-user/kafka/logs/server.
log"}},"message":"[2021-02-16 04:24:58,281] INFO [GroupCoordinator 3]: Preparing
 to rebalance group console-consumer-73430 in state PreparingRebalance with old
generation 0 (__consumer_offsets-48) (reason: Adding new member consumer-console
-consumer-73430-1-d8341695-331c-4c44-98bf-3b7141b33633 with group instance id No
ne) (kafka.coordinator.group.GroupCoordinator)","fields":{"pipeline":"kafka-logs
"}}
{"@timestamp":"2021-02-16T04:25:00.013Z","@metadata":{"beat":"filebeat","type":"
doc","version":"6.8.14","topic":"peter-log"},"offset":111543,"fields":{"pipeline
":"kafka-logs"},"beat":{"hostname":"ip-172-31-78-33.ec2.internal","version":"6.8
.14","name":"ip-172-31-78-33.ec2.internal"},"host":{"name":"ip-172-31-78-33.ec2.
internal"},"log":{"file":{"path":"/home/ec2-user/kafka/logs/server.log"}},"messa
ge":"[2021-02-16 04:24:58,286] INFO [GroupCoordinator 3]: Assignment received fr
om leader for group console-consumer-73430 for generation 1 (kafka.coordinator.g
roup.GroupCoordinator)","source":"/home/ec2-user/kafka/logs/server.log"}
```

심화

test-broker01/03 토픽에 쌓인 메세지 확인

- 빨간색으로 네모박스 친 ip가 각각 test-broker01/03의 AWS IP임
- test-broker01/03의 카프카 로그가 각 토픽에 잘 들어갔음을 확인할 수 있음

Chapter 5

느낀점

느낀점

END-TO-END 데이터 파이프라인 프로젝트

- 여러대의 AWS EC2를 구축하여 서로 통신하게 만들고 데이터 파이프라인을 처음부터 끝까지 짜봄
- 수집부터 간단한 시각화까지 모두 경험했다는 점에서 추후에 다른 데이터를 이용해서 심화된 프로젝트를 할 수 있는 가능성을 열어둠
- 파이프라인을 구축하며 여러 에러들에 직면했는데 그때마다 관련 로그를 봄으로써 해결하였음. 로그를 꼼꼼하게 보자는 교훈을 얻음

느낀점

추후 프로젝트 방향 & 부족한 점

- 현재는 데이터를 카프카 로그로 사용하였으나 다음 프로젝트에서는 크롤링 등으로 의미있는 데이터를 가져오고 싶음
- python application을 통해서 pub/sub을 구현할 수도 있지만 apache kafka가 java로 개발된 만큼, java을 공부할 필요가 있음
- kibana을 이용하여 멋진 dashboard을 구축하는 능력을 더 배양할 필요가 있어 보임