|  |
| --- |
| **Apache Kafka와 Elastic search를 이용한**  **Streaming Data Processing** |



**인공지능융합학과 2021123082 주민건**

**인공지능융합학과 2021116054 심유라**

**인공지능융합학과 2021123173 송우석**



1. **서론**

스트리밍(streaming) 데이터는 수천, 수만 개의 데이터 원본에서 연속적으로 생성되는 데이터로서 실시간 정보의 지속적인 흐름이자 이벤트 기반 아키텍처 소프트웨어 모델의 기반입니다. 현대적인 에플리케이션은 스트리밍 데이터를 사용하여 데이터를 처리, 저장, 분석합니다.

* 1. **Apache Kafka**

Apache Kafka는 대량의 실시간 스트리밍 데이터를 처리하는 가장 흔한 방법 중 하나로 자리 잡은 오픈소스 분산 메시징 플랫폼입니다. 실시간으로 기록 스트림을 게시, 구독, 저장 및 처리할 수 있는 분산형 데이터 스트리밍 플랫폼으로서 여러 소스에서 데이터 스트림을 처리하고 여러 사용자에게 전달하도록 설계되었습니다. 카프카는 데이터를 병렬로 처리함으로서 데이터를 빠르고 효과적으로 처리할 수 있습니다. Disk에 순차적으로 데이터를 적재하기 때문에 임의 접근(random access) 방식보다 훨씬 더 빠르게 데이터를 처리합니다. disk에 적재되기 때문에 만약 불의의 사고로 서버가 다운되었을 시에도 데이터가 유실되는 일 없이 재시작하여 기존 데이터를 안정적으로 처리 가능합니다. 또한, Scale-out이 가능하여 시스템 확장이 용이하며 어떤 하나 혹은 몇 개의 서버가 다운되더라도 서비스 자체가 중단될 일 없이 시스템 운용이 가능합니다.

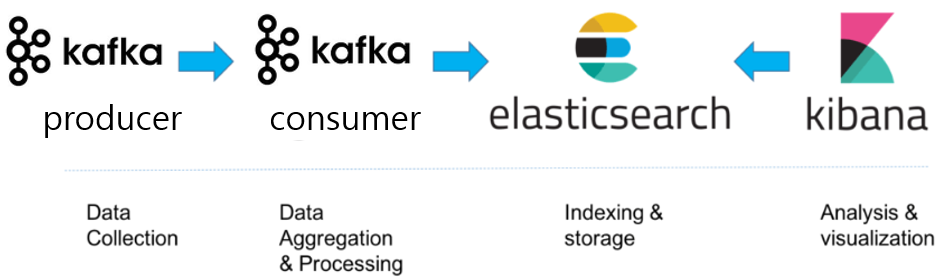
* 1. **Elastic Search**

Elastic search는 텍스트, 숫자, 위치 기반 정보, 정형 및 비정형 데이터 등 모든 유형의 데이터를 위한 무료 검색 및 분석 엔진으로 분산형과 개방형을 특징으로 합니다. 로그, 시스템 메트릭, 웹 애플리케이션 등 다양한 소스로부터 원시 데이터가 Elasticsearch로 흘러 들어갑니다. 데이터 수집은 원시 데이터가 Elasticsearch에서 색인되기 전에 구문 분석, 정규화, 강화되는 프로세스입니다. Elasticsearch에서 일단 색인되면, 사용자는 이 데이터에 대해 복잡한 쿼리를 실행하고 집계를 사용해 데이터의 복잡한 요약을 검색할 수 있습니다.

* + 1. **Elastic Stack** & **Kibana**

Elastic Stack은 데이터 수집, 시각화, 보고를 간소화합니다. Kibana는 시각화를 담당하는 HTML + Javascript 엔진입니다. Kibana에서 사용자는 데이터를 실시간으로 시각화하고, 대시보드를 공유하며, Elastic Stack을 관리할 수 있습니다. UI를 통해 애플리케이션 성능 모니터링(APM), 로그, 인프라 메트릭 데이터에 신속하게 접근할 수 있습니다.

본 프로젝트에서는 실시간으로 twitter에 글이 작성되는 상황을 다뤘습니다. Twitter가 작성된 시간과 이를 처리하는 machine의 id 정보가 포함된 일련의 숫자가 들어옵니다. 실시간으로 들어오는 스트리밍 데이터를 받아서 데이터에 포함된 유의미한 정보를 실시간으로 분석하는 시스템을 구축했습니다. 예를 들어 datetime과 machine id 정보를 활용하여 요일별, 시간대별 글 작성 수 또는 실시간으로 처리량이 많은 machine 찾기 등의 분석을 수행했습니다.



실시간으로 데이터를 처리하기 위해 Apache Kafka를 사용했고, kafka의 consumer가 받아온 데이터를 Elasticsearch라는 데이터관리시스템을 사용해서 데이터를 저장하고 분석한 내용을 자동으로 시각화하기 위해서 kibana라는 툴을 사용했습니다. Kibana를 사용해 실시간으로 들어온 정보를 요약하고 직관적인 dash board를 보여줄 수 있는 시스템 구축을 목표로 삼았습니다.

1. **데이터셋**

**Twitter streaming dataset on COVID-19 (2021 Nov)**

2021년 11월의 코로나-19와 관련된 모든 snowflake 아이디와 시간 데이터로

<https://stream.covid19misinfo.org/tweet_ids> 에서 다운로드 가능합니다. 본 프로젝트에서는 실시간으로 twitter가 작성된 시간과 이를 처리하는 machine id정보가 포함된 일련의 숫자가 들어오는 상황에 대해서 다뤘습니다. 총 데이터는 99403365개이며 19자리의 일련번호 형식입니다. 64bit로 변환했을 때 처음 41비트는 datetime 정보를 담고 있고, 그 뒤 10비트는 machine id 번호를 담고 있습니다. 본 프로젝트에서는 데이터로부터 시간과 machine id값을 찾고, 일을 많이 하는 machine과 요일별, 시간대별 작성된 글의 수를 실시간으로 보여주는 시스템을 구축하고자 했습니다.

1. **Kafka 시스템 설명 및 구현** 
   1. **Kafka**

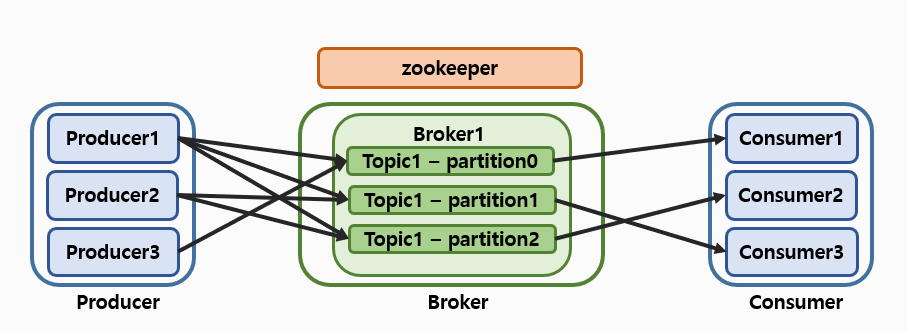
****

Figure 1

스트리밍 데이터 처리를 하기 위해 사용한 kafka의 구성요소는 다음과 같습니다.

* + 1. **메시지(Message)**

실시간으로 흘러 들어오는 데이터로 Snowflake id를 뜻합니다.

* + 1. **토픽(파티션)**

카프카에 저장되는 메시지는 Topic을 사용하여 분류되고(채널과 같은 역할) 한 topic은 여러 개의 partition으로 나뉠 수 있습니다. 메시지를 저장하는 버퍼와 같은 역할이라고 할 수 있습니다.

* + 1. **프로듀서(Producer)**

카프카에서는 메시지를 생산(produce)해서 카프카의 토픽으로 메시지를 보내는 역할을 하는 애플리케이션, 서버 등을 모두 프로듀서라고 부릅니다. 프로듀서의 주요 기능은 각각의 메시지를 토픽 파티션에 매핑하고 파티션의 리더에 요청을 보내는 것입니다. 프로듀서는 메시지를 보낼 때 메시지이 키값이나 라운드로빈을 사용하여 특정 partition으로 메시지를 전송합니다. 같은 키를 갖는 메시지는 같은 Partition에 저장하여 순서유지가 가능합니다. 라운드 로빈 스케줄링(Round Robin Scheduling,RR)은 시분할 시스템을 위해 설계된 선점형 스케줄링의 하나로서, 프로세스들 사이에 우선순위를 두지 않고, 순서대로 시간단위(Time Quantum)로 CPU를 할당하는 방식의 CPU 스케줄링 알고리즘입니다.

* + 1. **컨슈머(Consumer)**

프로듀서가 메시지를 생산해서 카프카의 토픽으로 메시지를 보내면 그 토픽의 메시지를 가져와서 소비하는 역할을 하는 애플리케이션, 서버 등을 지칭하여 컨슈머라고 합니다. 같은 컨슈머 그룹 안에 있는 컨슈머는 같은 partition에서 메시지를 받을 수 없습니다. 보통 partition 수와 consumer 수를 동일하게 구성하는 것이 일반적입니다.

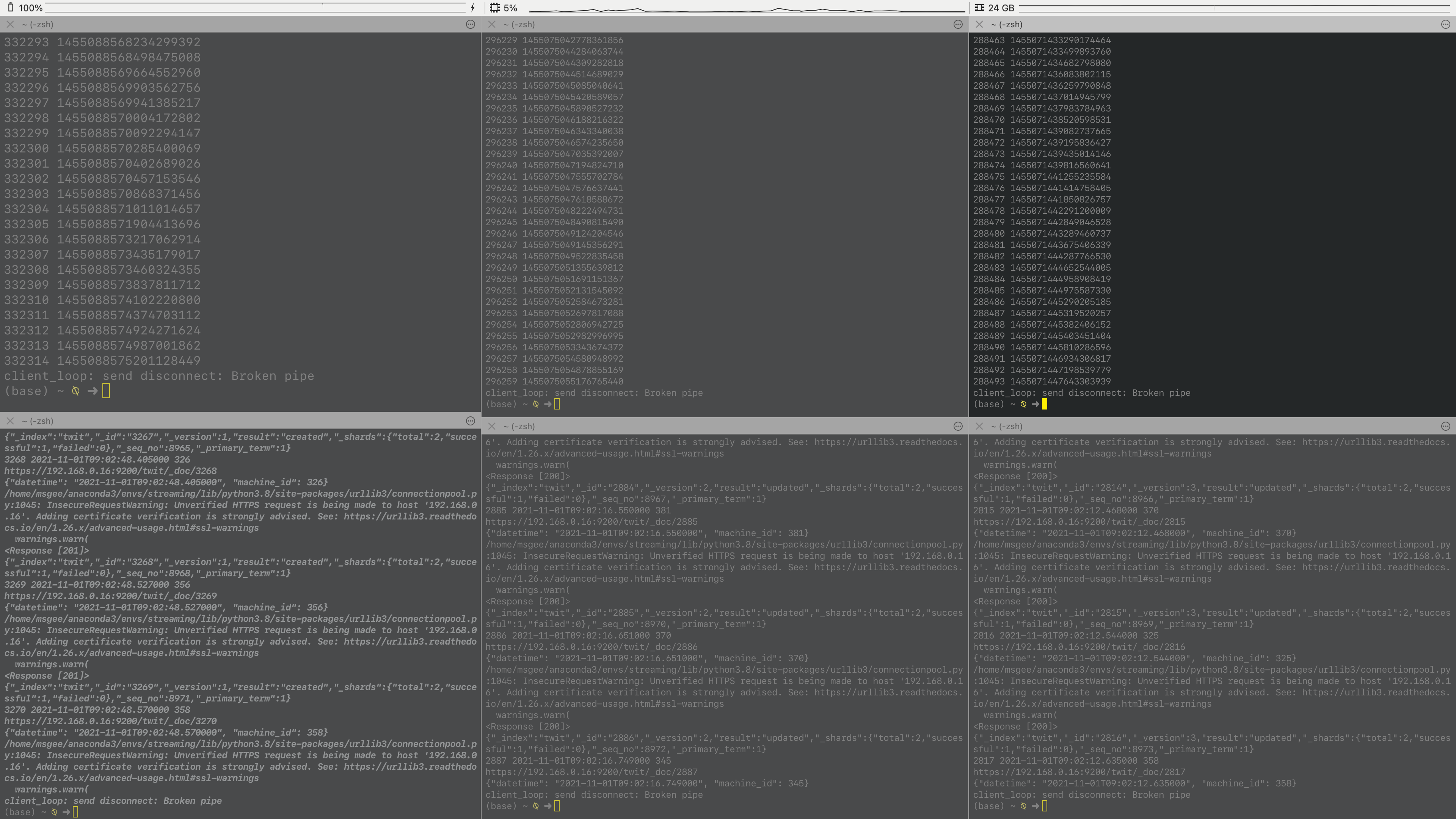


Figure 2. 위 3개 : Producer, 밑의 3개: Consumer

* + 1. **브로커(Broker)**

하나의 카프카 서버를 브로커라고 합니다. 브로커는 프로듀서로부터 메시지(Topic)를 오프셋(offset)을 지정한 후 해당 메시지를 디스크에 저장합니다. 브로커는 컨슈머 읽기요청에 응답하고 디스크에 수록된 메시지를 전송합니다. 하드웨어 성능에 따라 다르지만 초당 수천개의 토픽과 수백만 개의 메시지를 처리할 수 있습니다.

* + 1. **주키퍼(Zookeeper)**

카프카의 브로커를 관리해주는 소프트웨어로서 변경사항에 대해 카프카에 알리는 역할을 합니다. 예를 들어 Topic생성/제거, Broker 추가/제거 등을 알리며 주키퍼 없이는 카프카는 동작할 수 없습니다.

* 1. **Kafka setting 및 스트리밍 데이터 처리 과정**

본 과제에서 저희는 **Figure 1과 Figure2**에서 볼 수 있듯이 카프카 세팅을 Producer 3개, Consumer 3개, Broker 1개, Topic 1개(Partition 3개)로 구성한 후 세개의 producer 서버에 데이터가 임의로 나눠져서 들어오는 상황을 가정했습니다. 모든 데이터를 셔플한 후 세 개의 데이터셋으로 나누고 시간 순서로 정렬한 뒤, 실제 상황처럼 각 producer에 데이터를 흘려주었습니다. Producer는 처리되지 않은 snowflake id를 카프카 서버에 전송합니다. Consumer는 카프카 서버에서 읽어 온 snowflake id로부터 machine id와 datetime을 계산한 후(**Figure 3**) Elastic search로 전송하게 됩니다.

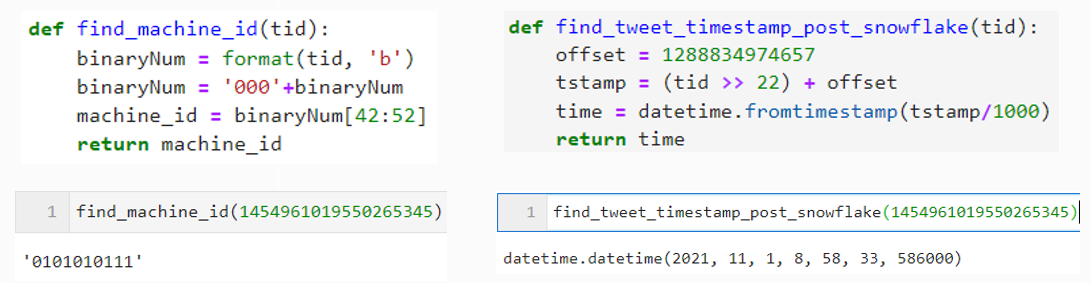


Figure 3. snowflake id : 1454961019550265345를 받아서 datetime과 machine id를 계산하는 example.

Elastic search에 데이터 인덱스 쿼리를 밑의 **Figure 4**와 같이 생성해줍니다.

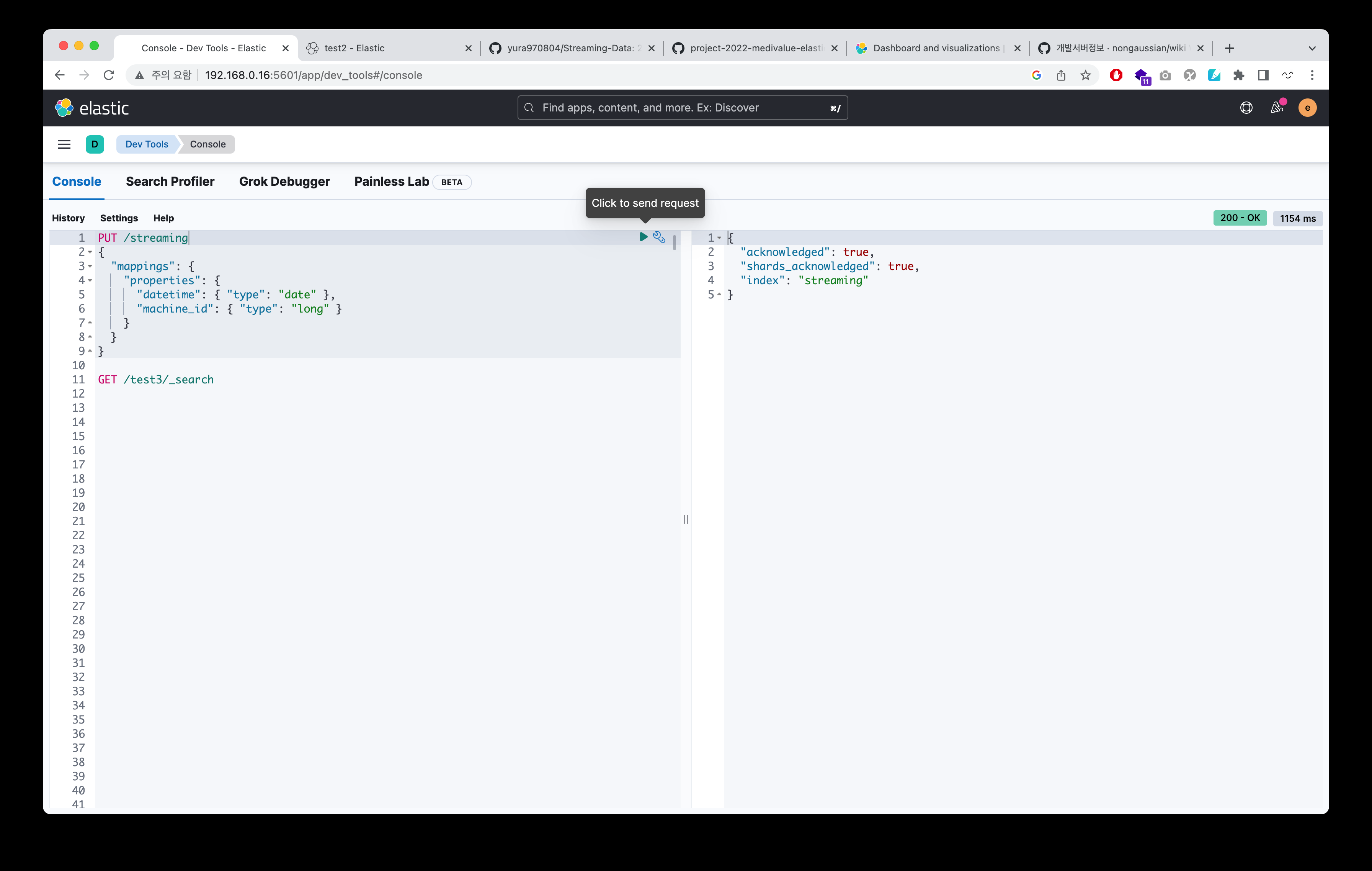


Figure 4. Elastic search에서 인덱스 쿼리를 생성하는 코드

Kibana에서는 Elastic Search에 저장된 datetime과 machine id 정보를 분석하여 시각화 해줍

니다**(Figure 5).**

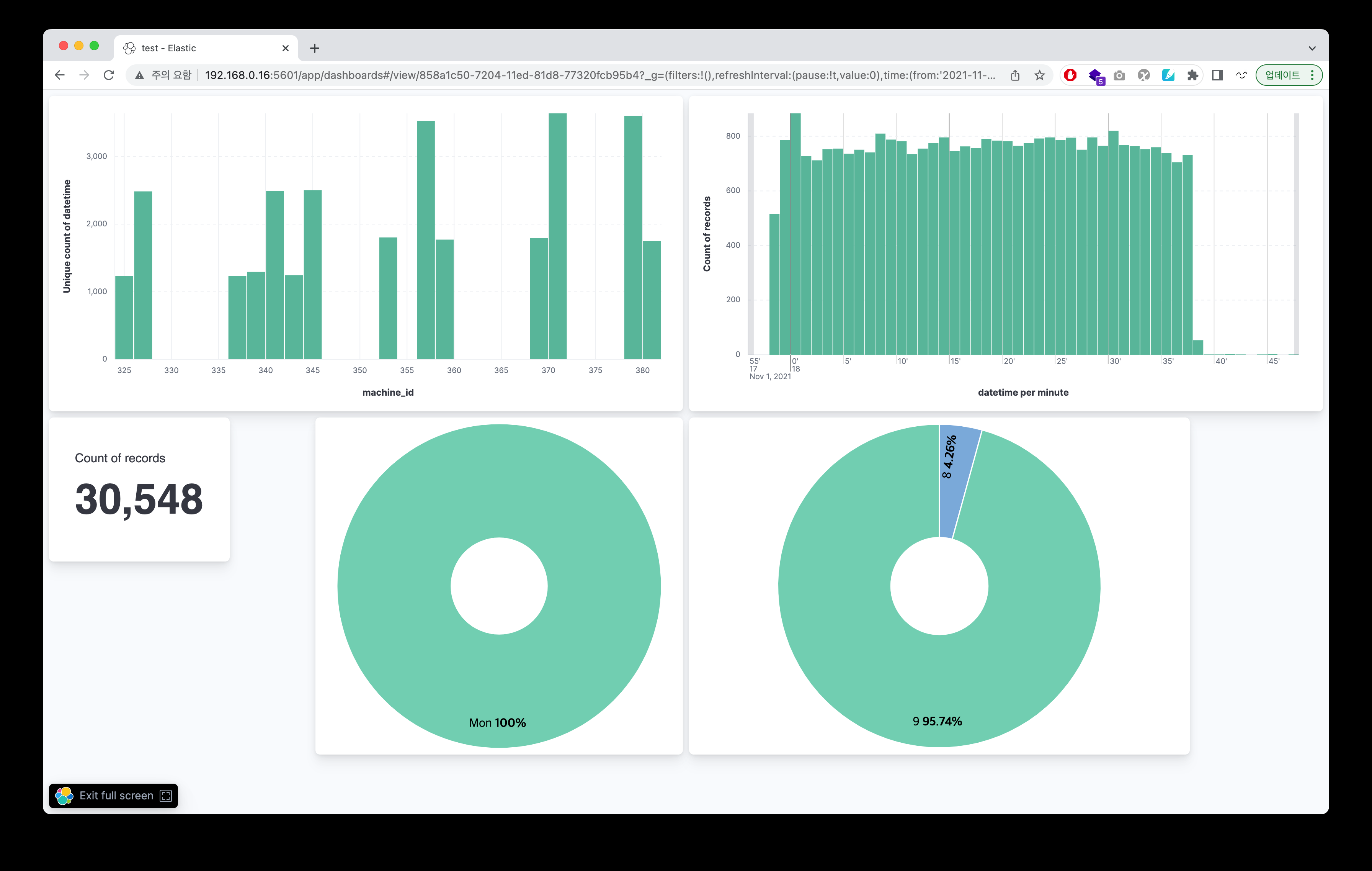


Figure 5

Figure 5는 실시간으로 들어오고 있는 데이터를 분석하여 시각화해주는 kibana tool을 보여줍니다. 왼쪽 아래의 숫자 30548은 현재까지 흘러 들어온 데이터 수를 나타냅니다. 왼쪽 위의 그래프는 machine id에 따른 빈도수, 오른쪽 위는 datetime per minute, 아래 두 그래프는 요일별, 시간대별 빈도수를 실시간으로 업데이트 해주고 있는 것을 알 수 있습니다. 총 약 9천만개의 데이터 중 시간순(11월1일 월요일)으로 약 3만개정도 데이터가 들어온 상태라 요일은 아직 월요일 100%인 것을 알 수 있습니다.

1. **결론 및 토의**

실시간으로 twitter 데이터가 흘러들어오는 상황을 가정하여 kafka와 Elastic Search를 사용해 데이터 분석을 해봤습니다. 카프카를 설치, Broker와 Zookeeper 실행, Topic 생성 및 관리, 메시지 주고받기 등을 하며 카프카를 다뤄볼 수 있는 프로젝트였습니다. 또한 Elastic search와 kibana툴은 실시간으로 분석된 데이터를 사용자가 편하게 볼 수 있어서 도움이 되었습니다.