# KEY VALUE 형태로 저장된 로그기록을 활용한 통계자료 보여주기

컴퓨터과학부 2015920056 최현호

## 1. 과제개요

### 1. 과제명과 팀원 소개

본 과제는 수업시간에 다룬 key value store 개념을 활용하여 로그 기록을 저장하고 통계자료를 보여주는 것입니다. 구체적으로는 채팅 웹을 만들고 채팅방마다 유저의 로그인 및 채팅 기록을 로그로 저장하고 시각적인 자료로 결과를 확인해 봅니다.

팀원은 최현호 1 인 팀이며 팀 명은 최현호입니다.

### 2. 과제 성립의 배경

웹사이트에서는 하루에도 수백 수천명의 사람들이 서비스를 사용하고 있습니다. 그러나 특정 서비스는 인기가 많고 특정 서비스는 거의 사용되고 있지 않습니다. 이러한 이용률의 차이는 서버 입장에서 자원 분배에 문제로 이어집니다. 사장된 서비스와 메인으로 이용되는 서비스에 동일한 자원을 분배한다면 남는 자원은 단순한 낭비로 볼 수 있지만 모자란 자원은 서버 다운의 원인이 됩니다. 그래서 로그 기록을 남겨 서비스 이용률을 감시해 변화를 지켜봅니다. 이 과정에서 막대한 양의 로그 기록이 생기기 때문에 단순한 형태로 저장한다면 탐색시간이 굉장히 오래 걸립니다. 탐색시간을 줄이기 위해서 key value store 형태로 로그를 저장하고 확인한다면 효율적인 서버 자원을 관리할 수 있습니다.

### 3. 설계대상의 기존 문제점 및 새로운 요구사항

본 과제는 기존의 없던 서비스의 개발이 아닌 key value store 의 효율적인 활용에 관한 서비스입니다. 다만 초기 MongoDB 에 로그기록을 저장하고 클라이언트에서 원할 때마다 확인해보는 방식으로 설계했습니다. 이후 실질적인 진행단계에서 매번 DB 에 접근하는 성능상에 문제점이 발생되었고 이는 ELK(Logstash + ElasticSearch + Kibana)라는 로그기록을 플랫폼을 이용하여 진행하였습니다.

프로젝트의 요구상항은

1. 실시간 채팅 웹을 구현한다.
2. 채팅 웹은 여러 개의 채널을 가져야한다.
3. 한 채널에는 여러 명의 유저가 입장할 수 있어야한다.

4. 실시간으로 로그를 기록해야 한다.

5. 로그 기록을 시각적으로 확인할 수 있어야한다.

### 4. 현실적인 제약 조건

본 서비스는 웹 서비스의 로그를 기록하기위해 서버를 구축해야 했지만 무료로 제공하는 서버에 한계가 있었습니다. 그래서 본 과제에서의 모든 서비스는 로컬에서 실행해야만 했습니다.

## 2. 설계과정

### 1. 세부목표 및 문제점

초기 설계에서는 모든 로그를 DB 에 저장하고 쿼리를 이용해서 시각화를 진행하려고 했습니다.

로그가 생기고 바로 DB 에 저장하게 되면 key value 가 구분이 되어있지 않아 일종의 String 형태로 저장이 됩니다. 그래서 key value 형태인 json 으로 저장하기 위해 로그 생성과 로그 저장 사이에 가공하는 단계가 필요했습니다. 그렇지만 이 과정이 채팅 서버에서 이루어진다면 수많은 로그마다 가공하는 과정이 포함되기에 서버에 부담이 될 수밖에 없습니다.

로그를 DB 에서 쿼리 형태로 불러와 시각적인 형태로 보여주는 단계에서도 문제가 생겼습니다. Html 을 이용하여 시각적인 자료를 만드는 과정에서 툴도 많지 않았고 특정 형태에 대에서만 그래프를 그려 재사용도 힘들었습니다.

### 2. 세부 목표에 해당하는 도출된 아이디어

위와 같은 세부 목표 및 문제점을 해결하기 위해서 여러 방법을 찾아보던 중 ELK 를 찾게 되었습니다. ELK 는 ElasticSearch, Logstash,Kibana 로 이루어진 오픈 소스입니다. ElasticSearch 는 검색 및 분석 엔진입니다. 이는 DB 에 해당하는 기능을 포함합니다. Logstash 는 여러 소스에서 동시에 데이터를 수집하여 변환한 후 ElasticSearch 같은 stash 로 전송하는 서버 사이드 데이터 처리 파이프라인입니다. Kibana 는 사용자가 ElasticSearch 에서 차트와 그래프를 이용해 데이터를 쉽게 시각화 할 수 있게 해줍니다. 이러한 3 가지 기능은 주로 한번에 묶여 ELK 라는 형태로 이용됩니다.

Logstash 를 이용하여 서버에서 매번 데이터 가공하는 과정을 줄여 편의성과 자원을 아낄 수 있었습니다.

ElasticSearch 는 일반적인 DB 와 달리 역 인덱스 형태로 데이터를 저장하여 데이터의 접근시간을 매우 많이 줄여줍니다.

| 텀(Term) | ID | 텀(Term) | ID |
|---|---|---|---|
| The | doc1, doc2, doc3 | quick | doc1, doc2, doc3 |
| brown | doc1, doc2, doc3, doc4 | fox | doc1, doc2, doc3, doc4 |
| jumps | doc2, doc3 | over | doc2, doc3 |
| the | doc2, doc3 | lazy | doc2 |
| dog | doc2, doc3, doc4, doc5 | Brown | doc4 |
| Lazy | doc5 | jumping | doc5 |

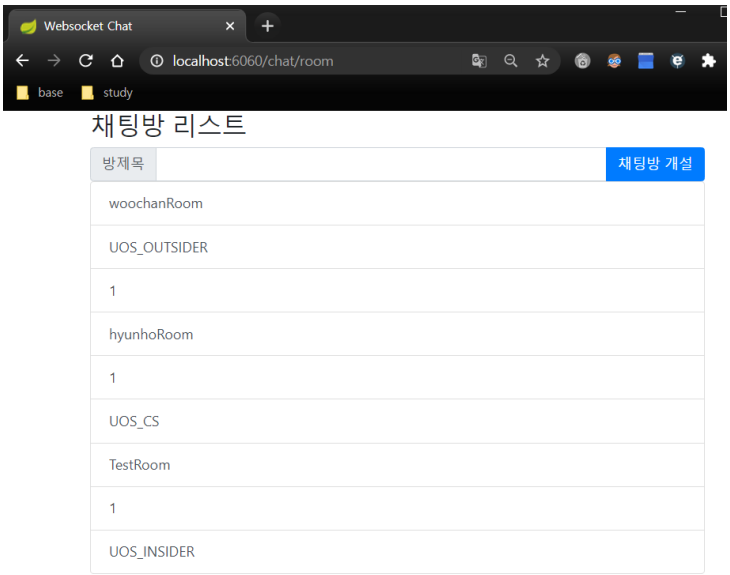
Kibana 는 여러가지 시각화 툴을 제공하여 ElasitcSearch 와 사용하기에 적절하였습니다.

## 3. 최종 설계 결과물

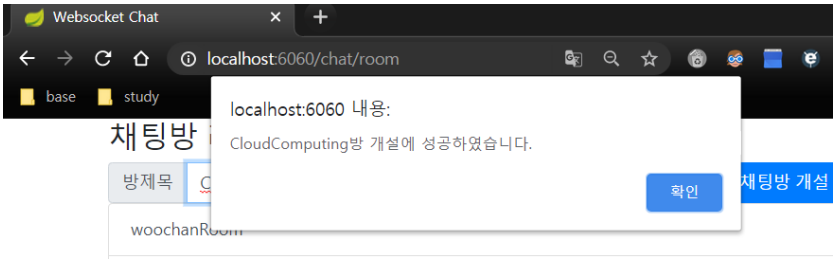
### 1. 최종 설계 결과물 제시

먼저 최종적인 결과를 사진으로 보여드리겠습니다.

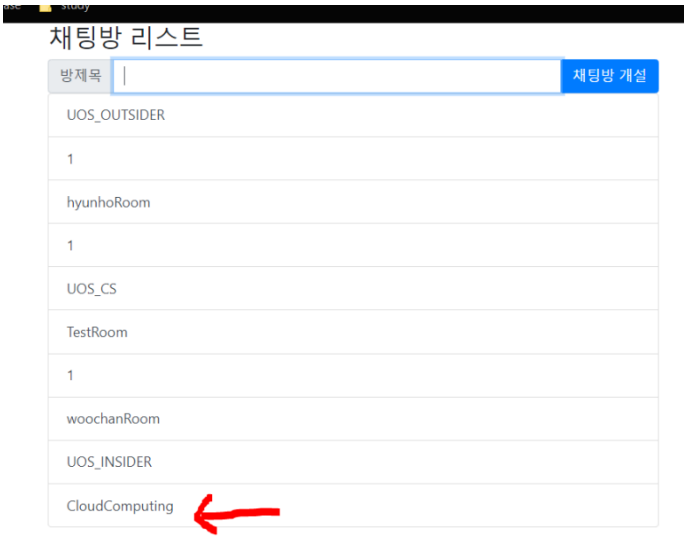
클라이언트가 접속한 채팅 웹 프론트 화면입니다. 채팅 방 리스트가 나오게 되고 채팅 방 제목을 입력하고 채팅 방 개설을 누르게 되면 새로운 채팅방이 생성됩니다.




<채팅 웹 메인 화면>

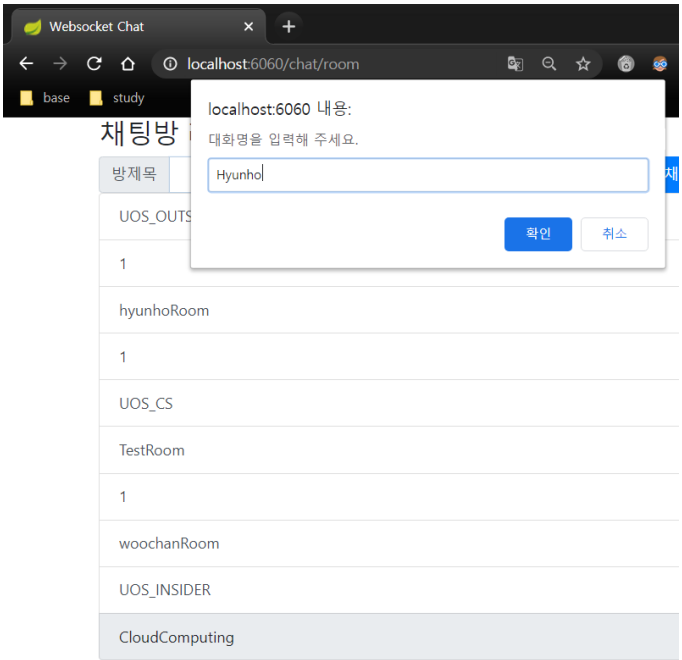



<채팅 방 개설>



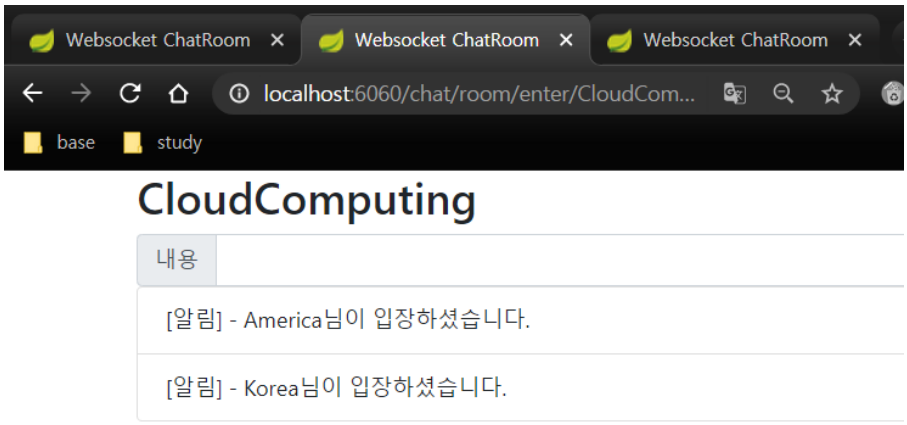

<개설된 채팅 방>

채팅방이 개설되면 개설된 채팅방을 눌러 입장을 할 수 있습니다. 채팅방을 누르면 유저의 닉네임을 설정하고 확인을 눌러 입장합니다.



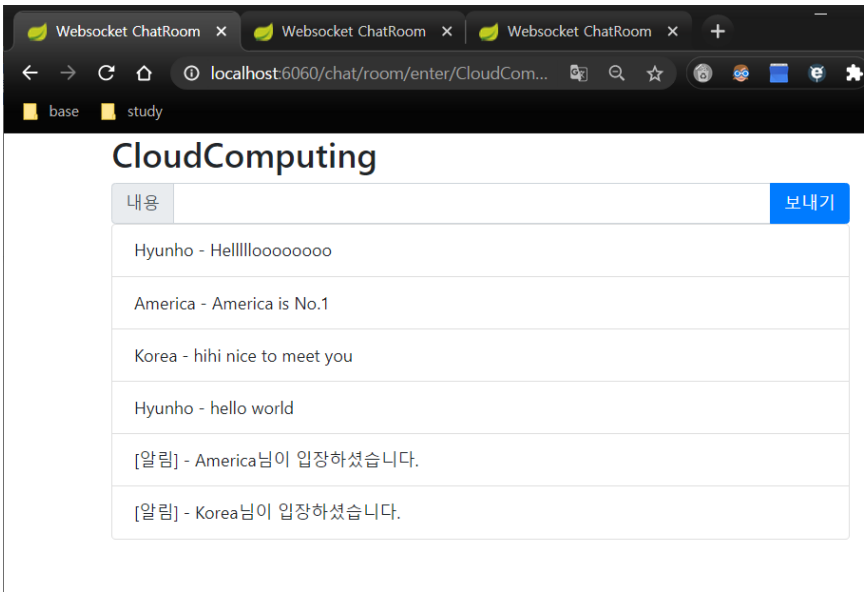

<채팅 방 입장>

새로운 세션을 열어주고 여러 닉네임으로 입장이 가능합니다.



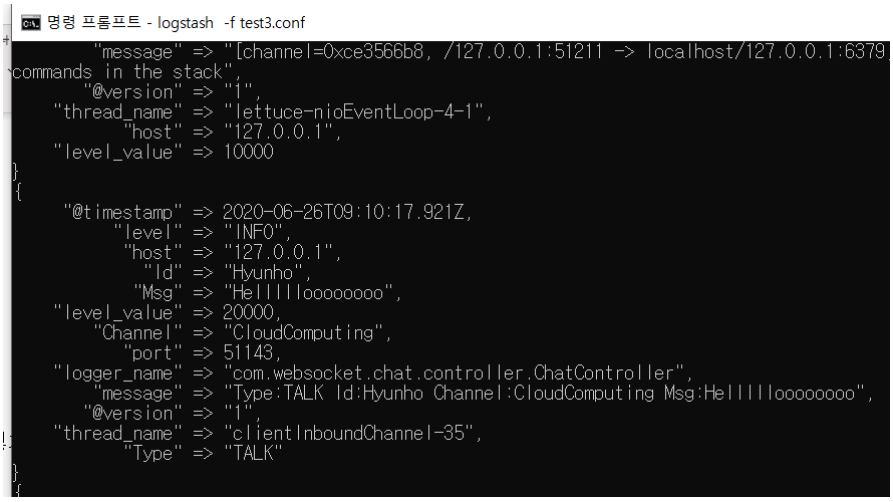

<닉네임 Korea, America 로 입장>

생성된 유저마다 채팅을 입력하면 채팅이 기록이 됩니다.

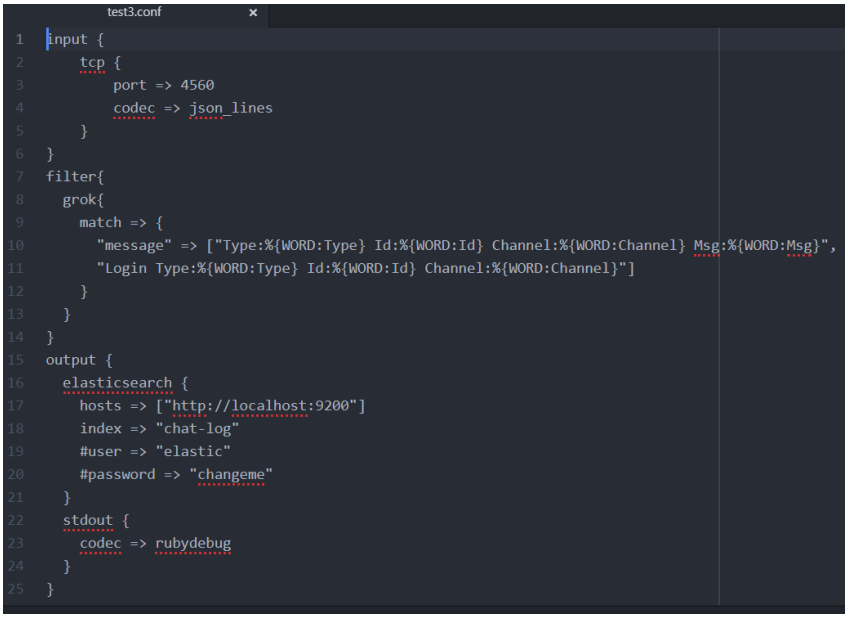



<기록되는 채팅>

이렇게 기록된 채팅은 logstash 와 연결하여 json 형태로 로그가 변환됩니다.




<logstash 에서 json 형태로 가공>




어떤 형식으로 저장할지 어떠하게 가공할지는 config 파일에서 설정하였습니다.

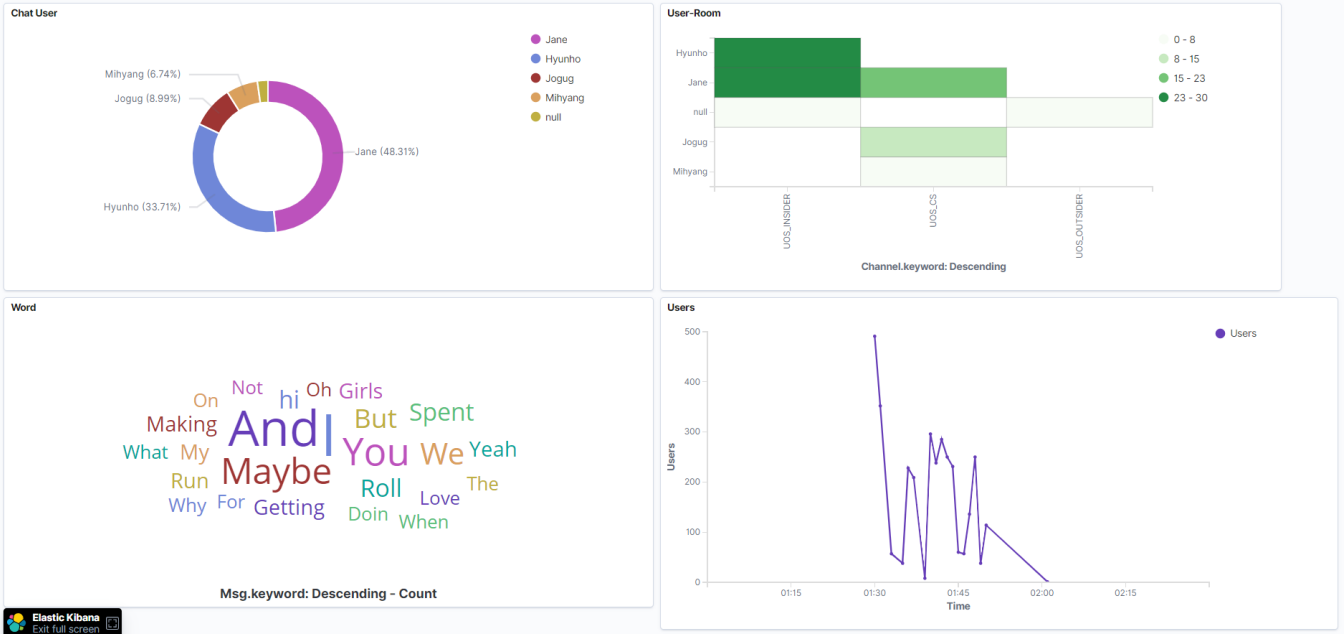
Logstash 를 통해서 실제로 로그가 저장되는 곳은 ElasticSearch 입니다.

<실행 되어있는 ElasticSearch>

최적으로 개발자는 kibana 의 UI 를 통해 로그가 기록된 것을 확인합니다.


<Kibana 에서 확인 가능한 로그 기록>

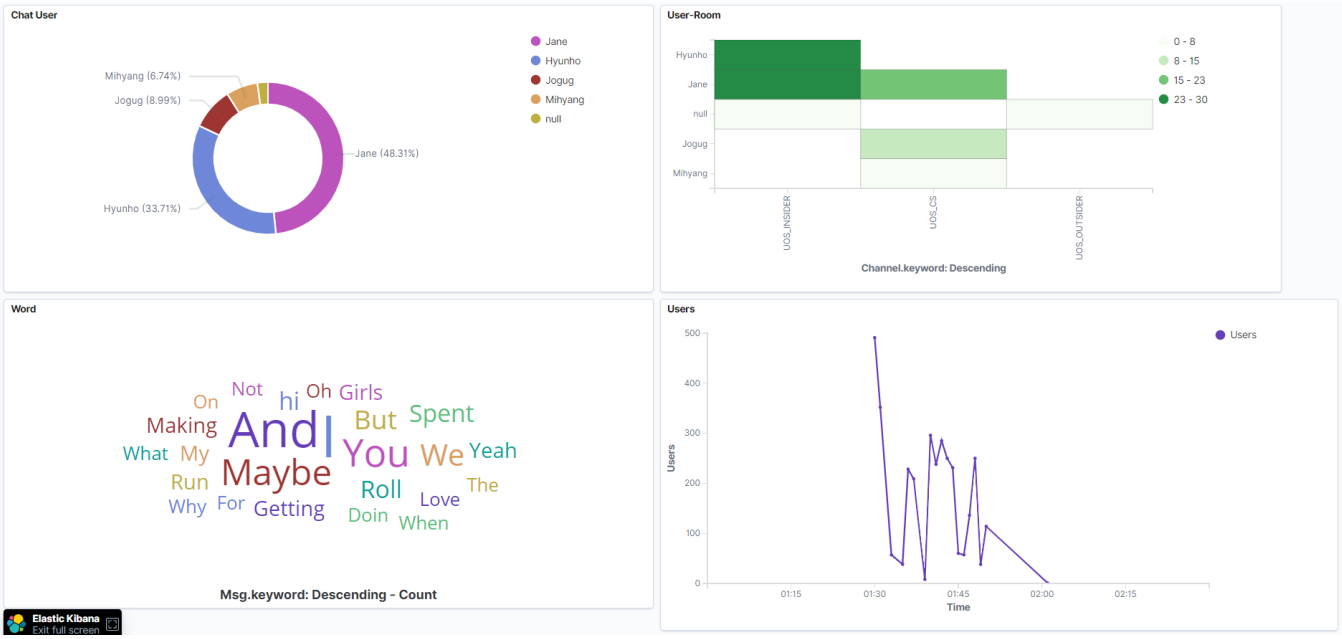
이제 생성된 로그를 데이터로 하여 여러가지 분석 그래프를 그려봅니다. 이는 Kibana 에서 제공하는 툴을 이용한 그래프입니다.




<kibana 를 활용한 그래프>

## 결과물의 성과

최종적으로 확인할 결과는 로그 기록을 기반으로 생성된 그래프입니다. 그래프를 해석함으로써 서버 관리 및 유저 관리에 이용할 수 있습니다. 위에서 생성한 그래프를 예시로 설명해보겠습니다.



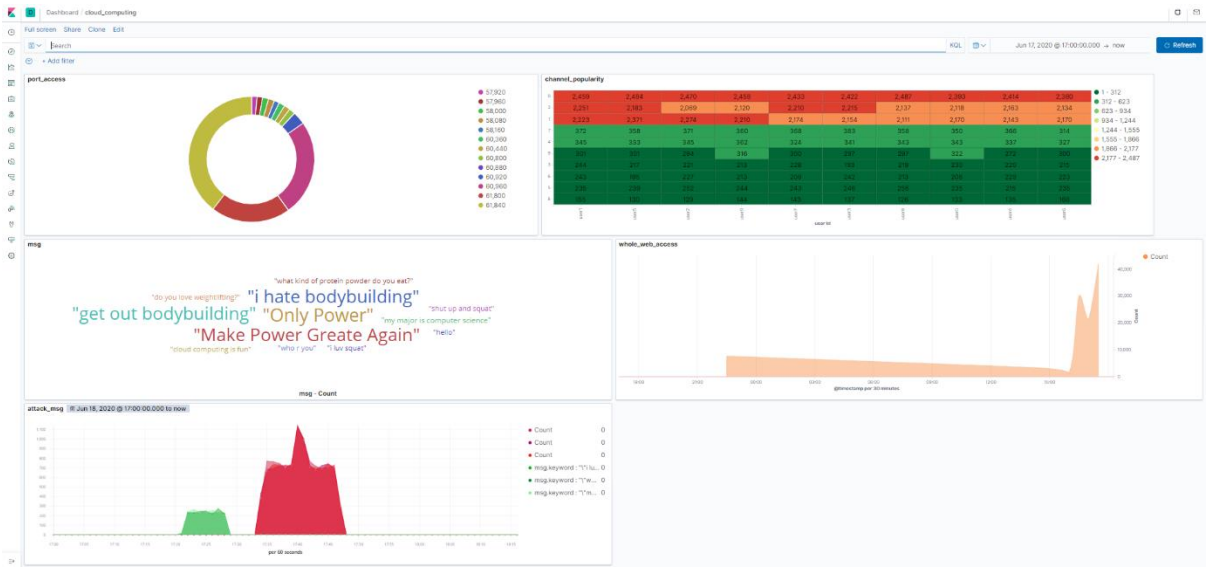

<kibana 를 활용한 그래프>

왼쪽 위쪽부터 오른쪽 위, 왼쪽 아래, 오른쪽 아래 그래프 1,2,3,4 로 명명하겠습니다. 그래프 1 은 채팅 유저의 로그 기록 비율을 나타낸 것입니다. 이를 통해 어떤 유저가 가장 활발하게 사용하는지 알 수 있습니다. 그래프 2 는 채팅 채널과 유저로 분석한 채팅 채널 혼잡 비율입니다. 그래프를 보면 한쪽 채널에는 사용자 빈도수가 많고 반대쪽 채널은 거의 이용되지 않는 것을 확인해볼 수 있습니다. 그래프 3 은 워드 클라우드입니다. 많이 나온 단어를 분석한 데이터 사이언스에서 활용할 수 있습니다. 그래프 4 는 시간당 채팅 이용률입니다. 특정 시간대에만 유저가 몰리는 것을 확인할 수 있습니다.

## 설계 결과물 및 작품 평가

위의 예시는 단순히 수십개의 채팅을 이용한 예시입니다. 이는 실제 채팅 로그 기록의 비해 터무니없이 작은 양이어서 만든 서비스를 평가하기에 부적절하다고 판단하였습니다.

그래서 실제와 같은 비슷한 환경을 만들어보고 위해 매크로를 만들어 대량의 채팅기록을 테스트해보았습니다.




<테스트 결과물>

약 10 만여개의 채팅 로그 기록을 만들어보았습니다. 데이터가 많아지니 그래프의 형태가 점점 자연스러워졌습니다. 2 번째 그래프를 확인해보면 채널 0,1,2 에만 거의 모든 사용자가 몰려 있는 것을 확인하였고 서버에서는 채널 0,1,2 에 자원을 더 할당하던지 제한을 둬 자원의 비대칭을 막을 수 있습니다. 또한 일부러 특정 시간대에 특정 키워드를 연속에서 남발된 것이 포착되었습니다. 그래프 5 번을 확인해보면 초록색 부분이 일반적인 키워드이고 빨간색 부분이 갑자기 많이 나타난 키워드입니다. 이러한 구체적인 분석을 통해 악성 유저 또한 걸러낼 수 있음을 확인하였습니다.

## 결론

대량으로 발생하는 로그 기록을 체계적으로 저장하고 그래프로 확인할 수 있었습니다. 이를 통해 서버를 관리하는 개발자는 업데이트 되는 로그 그래프를 확인하며 서버의 어떠한 문제가 발생하는지를 확인하고 기술적이고 비용적인 문제를 해결할 수 있습니다. 유지보수가 개발에 포함되는 부분을 생각했을 때 본 서비스는 프로젝트의 업무를 효율적으로 처리할 수 있게 해줍니다.

하지만 본 프로젝트에서는 모든 문제를 보여주는 것이 아닌 대량의 문제가 발생했을 때만 지표로 확인할 수 있어 다른 유형의 문제는 발견하지 못합니다. 그리고 문제가 발생했다고 알려만 줄 뿐 해결에 대한 방법은 안내하고 있지 않습니다. 또 문제가 발생했는지는 그래프를 보고 사용자가 직접 문제인지 판단해야 하는 한계점이 있습니다.

추가적으로 로그 기록의 형태에 따라 logstash 에서 필터를 매번 새롭게 설정해야 한다는 점, 로그 형태에 따라 그래프 또한 매번 새롭게 그려줘야 한다는 점, 재정적인 문제로 인해 모든 과정을 로컬에서 진행했다는 점이 아쉽습니다. 문제 1,2 번은 추가적인 툴을 만드므로써 해결할 수 있을 것으로 판단합니다. 문제 3 번은 재정적인 문제만 해결이 된다면 실제 서비스를 런칭해 보고싶습니다.