2021 Woori Digital Academy
Digital Analytic Master (DAM) 과정
프로젝트 결과보고서

(Financial Apps Review Analytics 서비스 포함)

목 차

01 DAM 과정 소개
과정 내용

02 실습 및 과제 수행
서비스 구현 프로젝트 수행

DAM 과정 소개

Digital Analytic Master

과정 내용

과정 내용(약식)

✓ Hadoop
- HDFS
- MapReduce
- Flume
- Sqoop
- Hive

✓ Spark
- Spark RDD
- Spark SQL
- Spark Streaming(mini-batch, structured)
- Spark MLlib(Machine learning)

✓ Kafka

✓ Real-time Streaming

실습 및 과제 수행

과정 내용을 토대로 한 데이터 서비스 구현

Financial Apps Review Analytics

프로젝트 목표

금융 앱 강,약점 분석

✔국내,외 금융사 앱 리뷰의
긍/부정 분류를 통한 강/약점 분석

✔Hadoop의 HDFS/MapReduce를 통한
Word Count 처리 및 Rank

Demand 파악 및 Notification/Report

✔앱 리뷰 언급 횟수를 통해
사용자 요구를 카테고리 분류/보고서 서비스

✔서비스 담당자에게 Notification 지원

프로젝트 주제 선정 배경

핀테크 앱, 금융뱅킹앱 추월…토스·카뱅 앱 접속건수 1·2위

한국 소비자들은 지난해 금융 앱을 약 410억회 방문했고, 일주일 평균 은행 월렛 앱은 4.6회, 핀테크 월렛 앱에는 11.7회 방문한 것으로 나타났다.

뱅크 월렛 앱은 1위가 KB스타뱅킹, 2위 NH스마트뱅킹, 3위 신한 쏠, 4위 우리은행 원터치개인뱅킹, 5위에는 신한페이판이 올랐다.

애플, 아마존에 이어 구글 등 빅테크 기업이 금융 분야로의 진출에 속도를 내고 있다.

23일 외신 등에 따르면 구글은 지난 18일(현지시간) 은행 계좌 개설과 간편송금, 생활비 관리 등의 기능이 추가된 '구글 페이'를 출시했다.

핀테크, 인터넷은행의
시장점유 증가

시중은행의 비대면 서비스 확대

빅테크 기업의 금융 진출

우리금융의
MAU 증대를 위한
비대면 서비스 활성화 전략 필요

구축 Preview

개발 Workflow

개발환경

아키텍쳐

시스템 구성도

주요기능소개

개발
Workflow


데이터 수집
Play스토어, 앱스토어 크롤링
4대 은행 앱
인터넷은행 앱
핀테크앱(토스 외)
8대 카드 앱
은행의 각종 앱 리뷰 수집
전처리
자연어 분석
단어 토큰화
불용어 정의
불용어 처리
이상치 제거
데이터 시각화
워드 클라우드
그래프
모델 학습
긍/부정 분류기
SVM
KNN
RandomForest
기능 상세 분류기
LSTM
GRU
모델 평가
Precision
Recall
F1 Score
ROC Curve
Accuracy
Loss
빈도분석
HDFS
WordCount
Spark
Hadoop
Service
Push-Slack
WebService

개발환경

Server Instance

Managing Source Codes

IDEs

KMS, Wiki, Managing Project

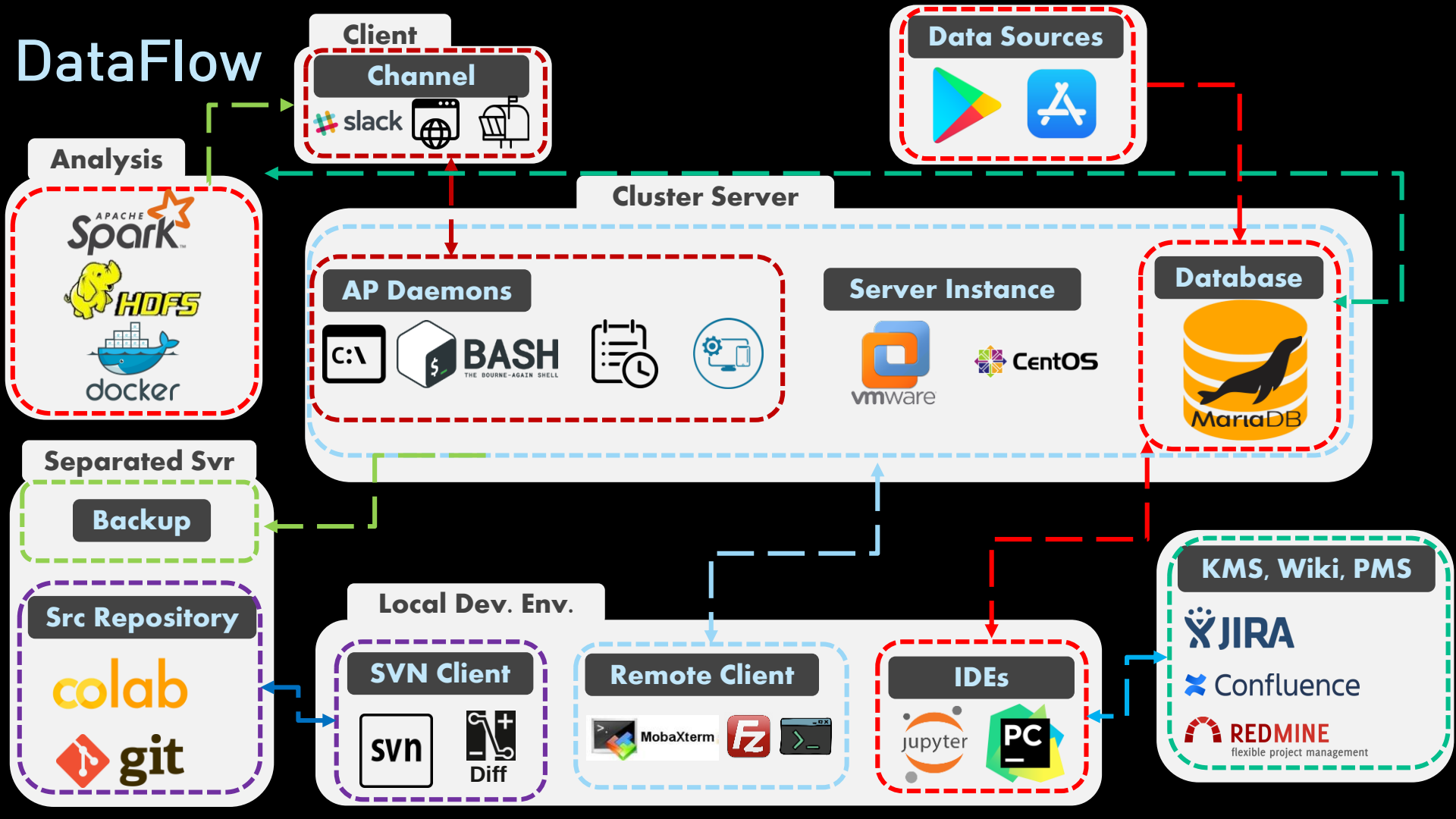
Languages

PIMS

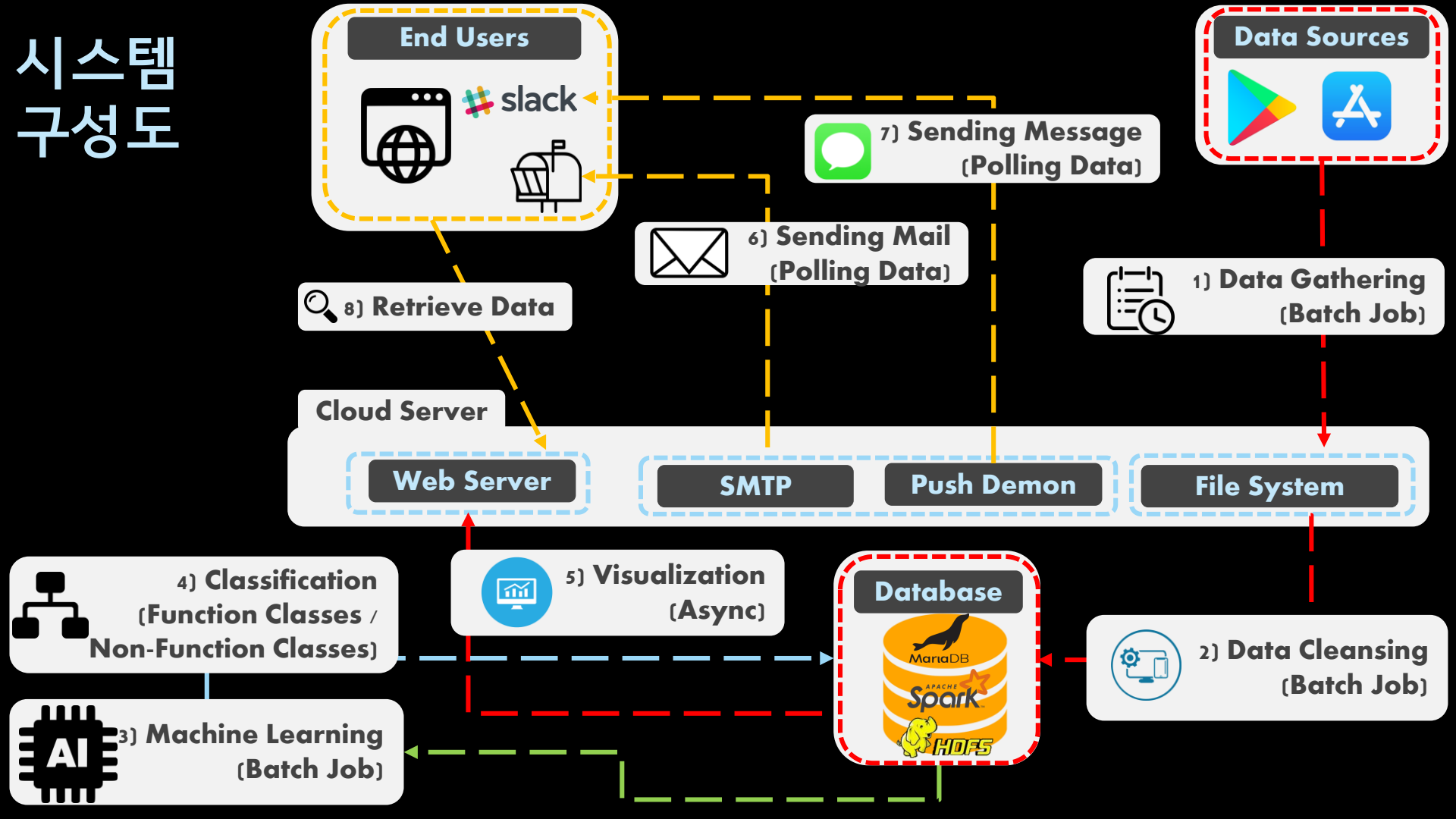
Database

Data Sources

Channel

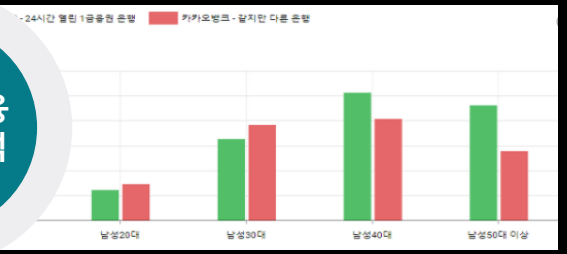

DataFlow
Client
Channel
slack
Data Sources
Analysis
APACHE Spark
HDFS
docker
Cluster Server
AP Daemons
C:\
BASH
THE BOURNE-AGAIN SHELL
Server Instance
vmware
CentOS
Database
MariaDB
Separated Svr
Backup
Src Repository
colab
git
Local Dev. Env.
SVN Client
svn
Diff
Remote Client
MobaXterm
IDEs
jupyter
PC
KMS, Wiki, PMS
JIRA
Confluence
REDMINE
flexible project management

시스템 구성도

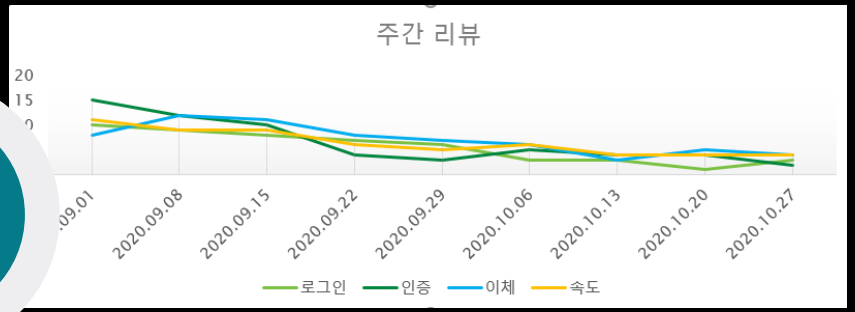

End Users
slack
Data Sources
7) Sending Message (Polling Data)
6) Sending Mail (Polling Data)
8) Retrieve Data
1) Data Gathering (Batch Job)
Cloud Server
Web Server
SMTP
Push Demon
File System
4) Classification (Function Classes / Non-Function Classes)
5) Visualization (Async)
Database
MariaDB
Spark
HDFS
2) Data Cleansing (Batch Job)
AI
3) Machine Learning (Batch Job)

주요기능 소개

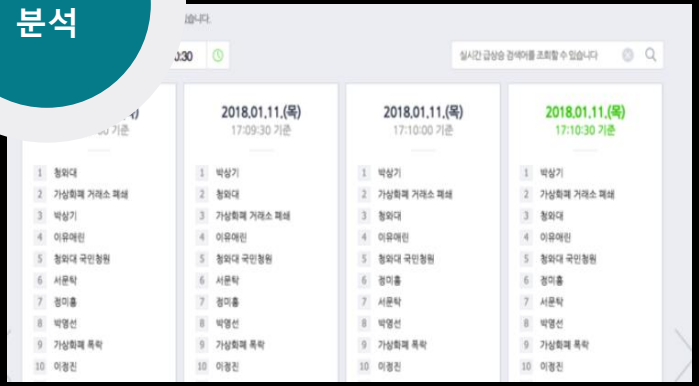
1.우리금융 비교분석

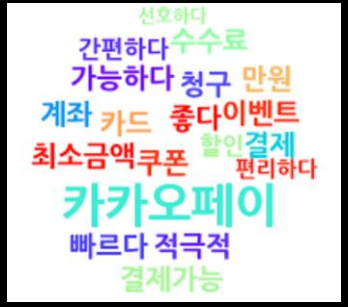


2.금융사별 상세분석



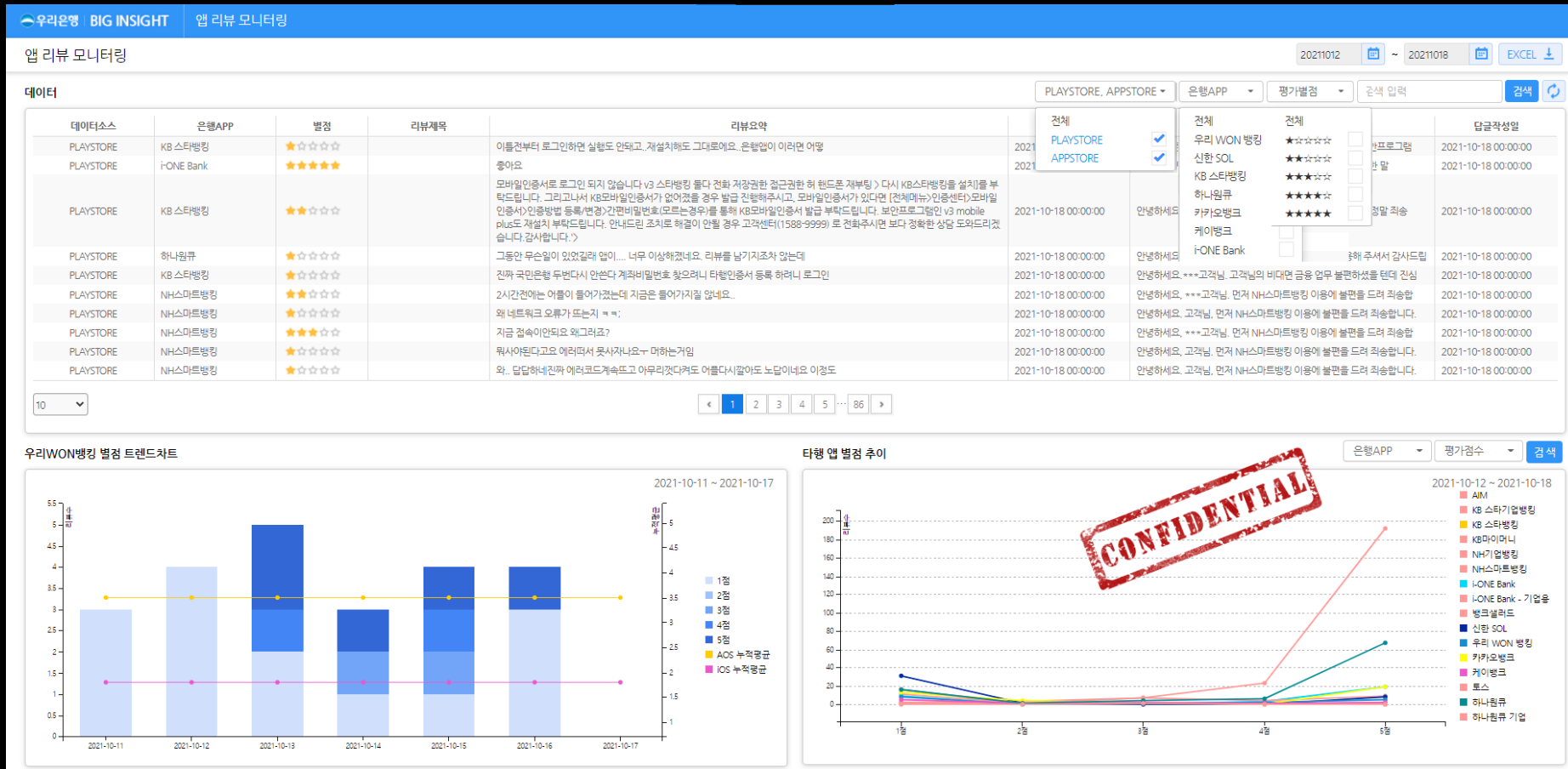
3.다빈도 단어 분석





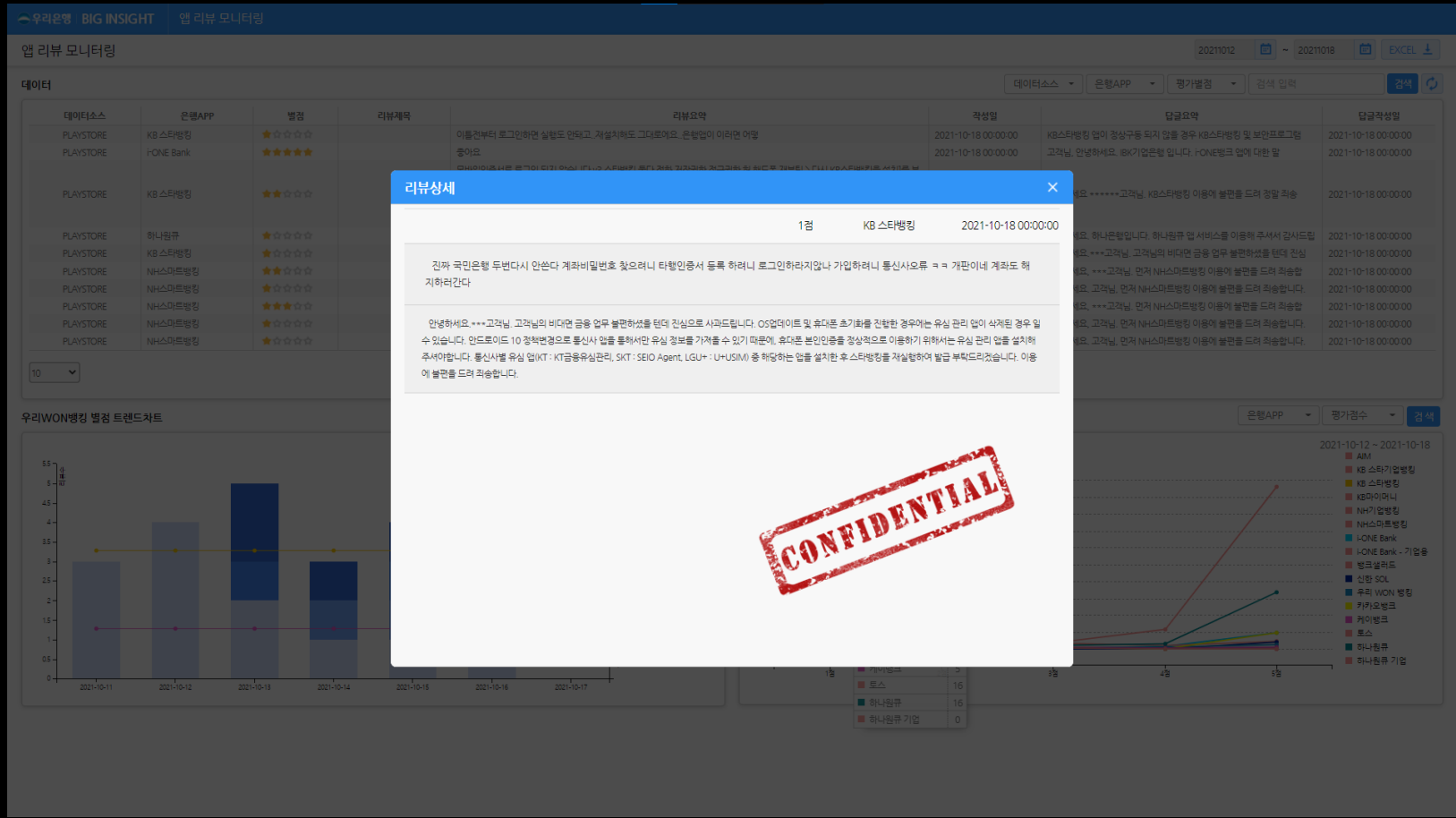
(비고)그룹 내 관련서비스(앱리뷰모니터링)

1.Graph/Filter/DashBoard



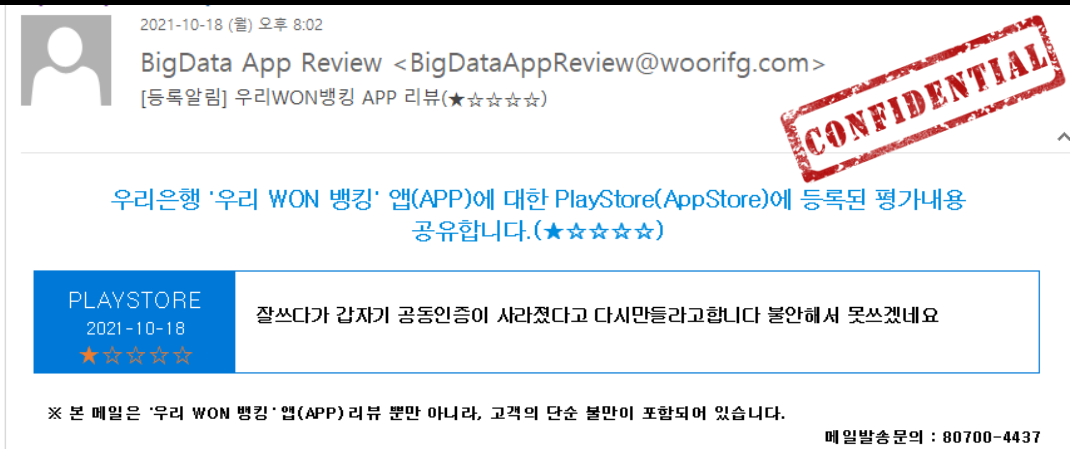
(비고)그룹 내 관련서비스(앱리뷰모니터링)

2.Review Text Contents



(비고)그룹 내 관련서비스(앱리뷰모니터링)

3. Internal Email Push

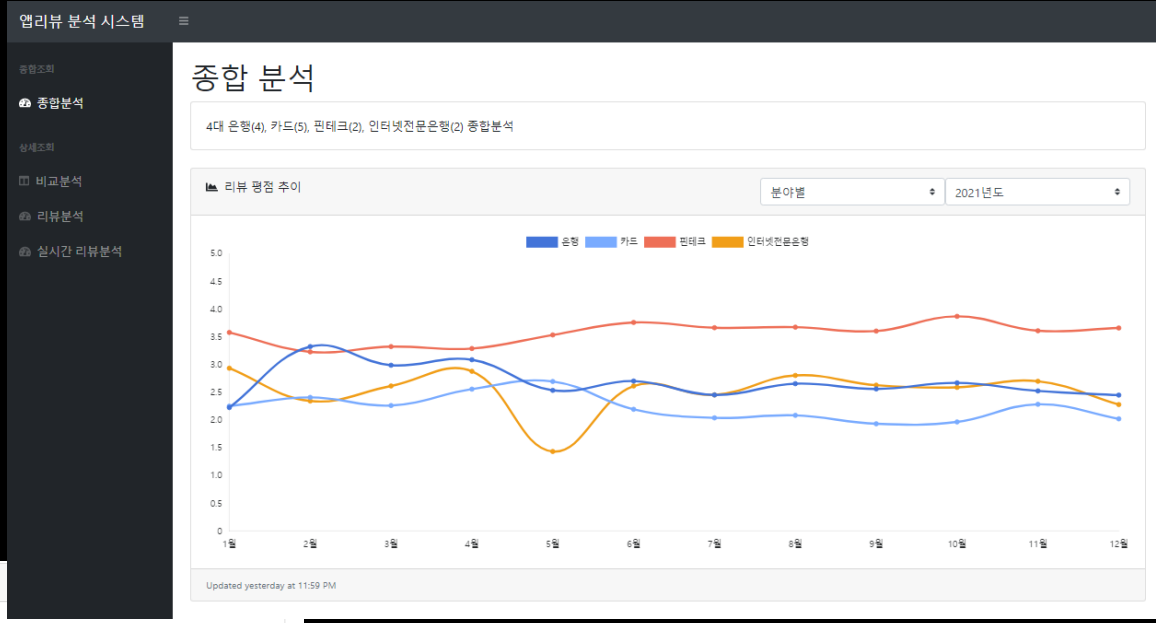


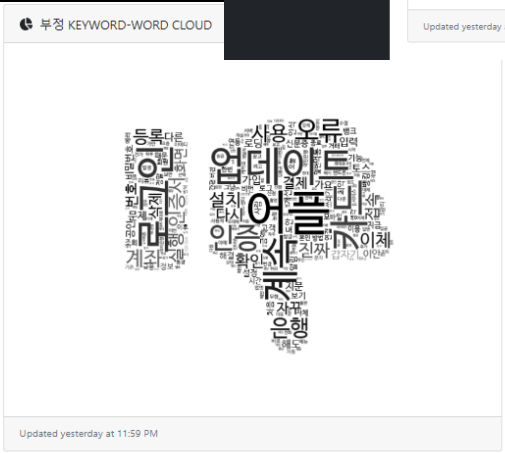
구축 서비스 소개

WebService / Push / Batch / Etc

Web Service

Report(**평점 변화 추이, 긍/부정 Words**)
http://kaydash.iptime.org

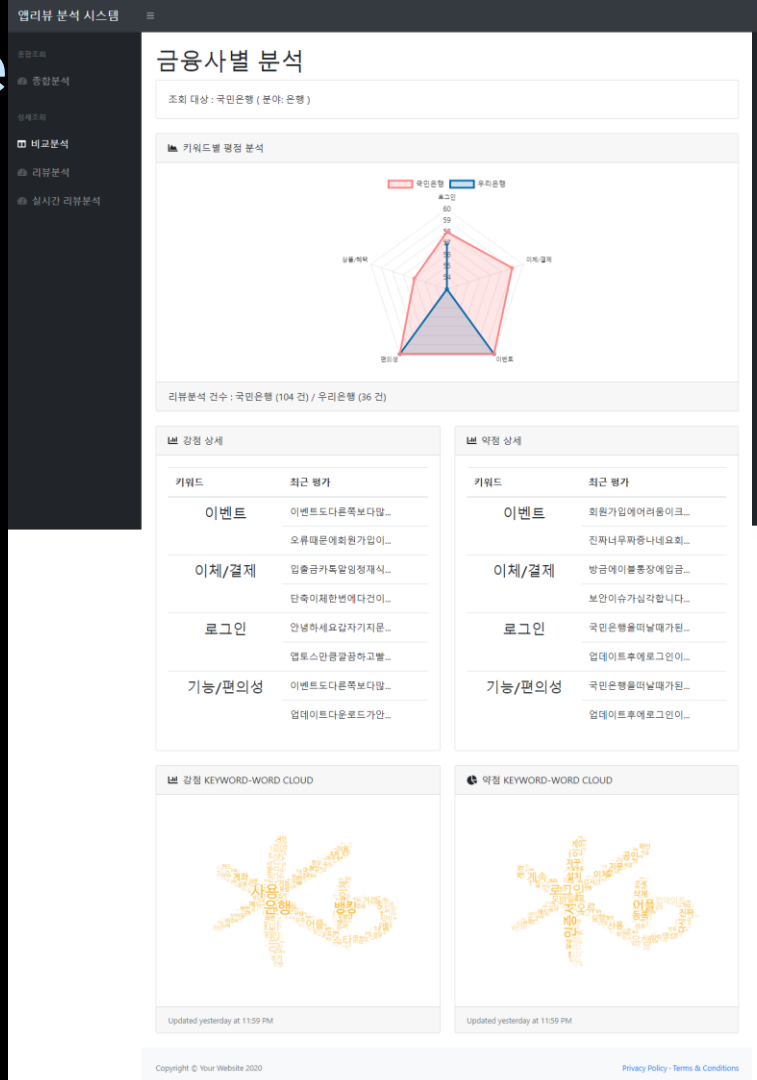


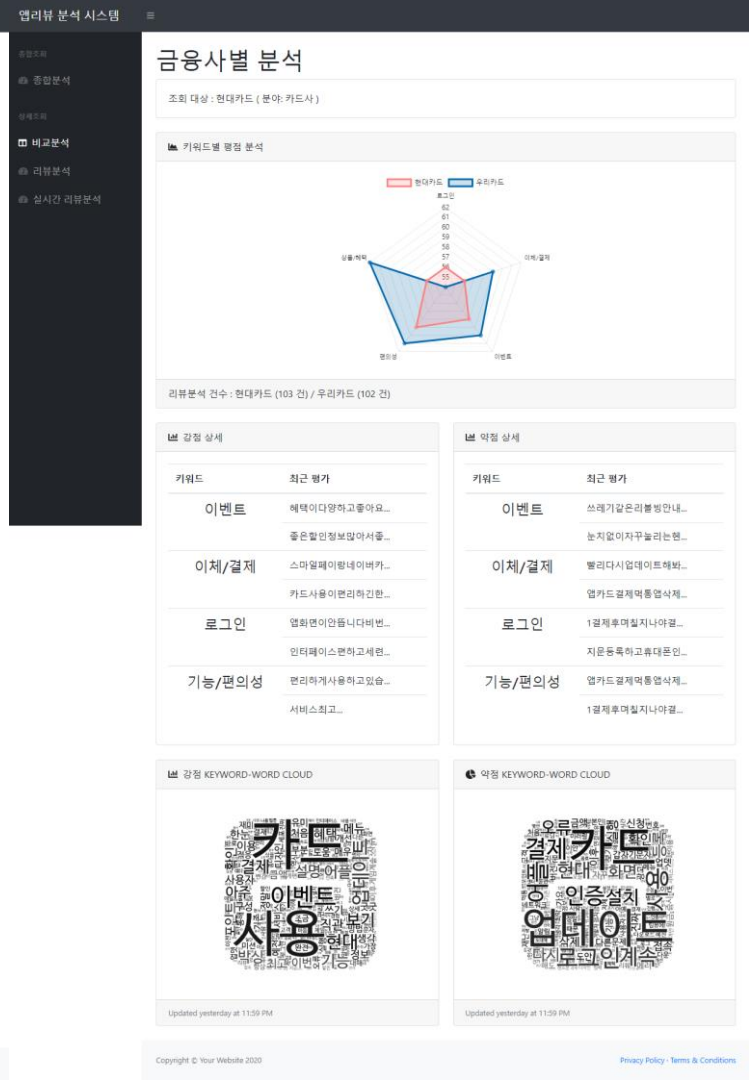


Web Service

Differencial Analytics

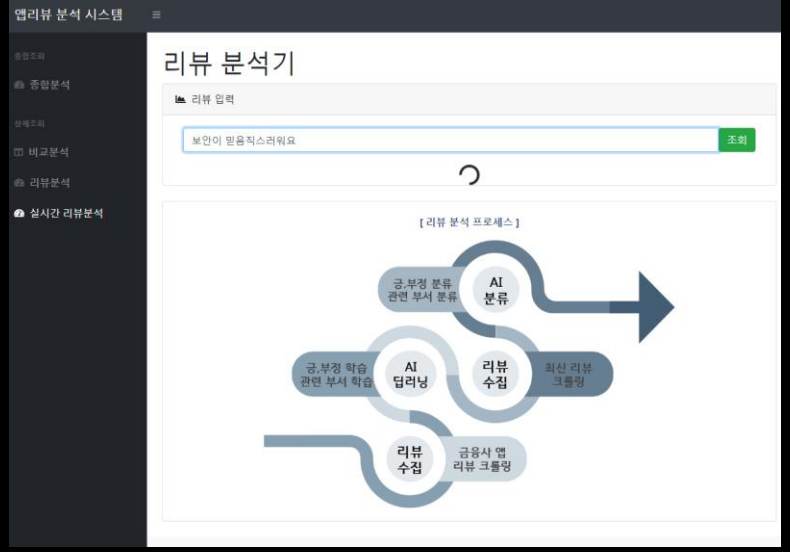
우리은행 vs 국민은행
우리카드 vs 하나카드





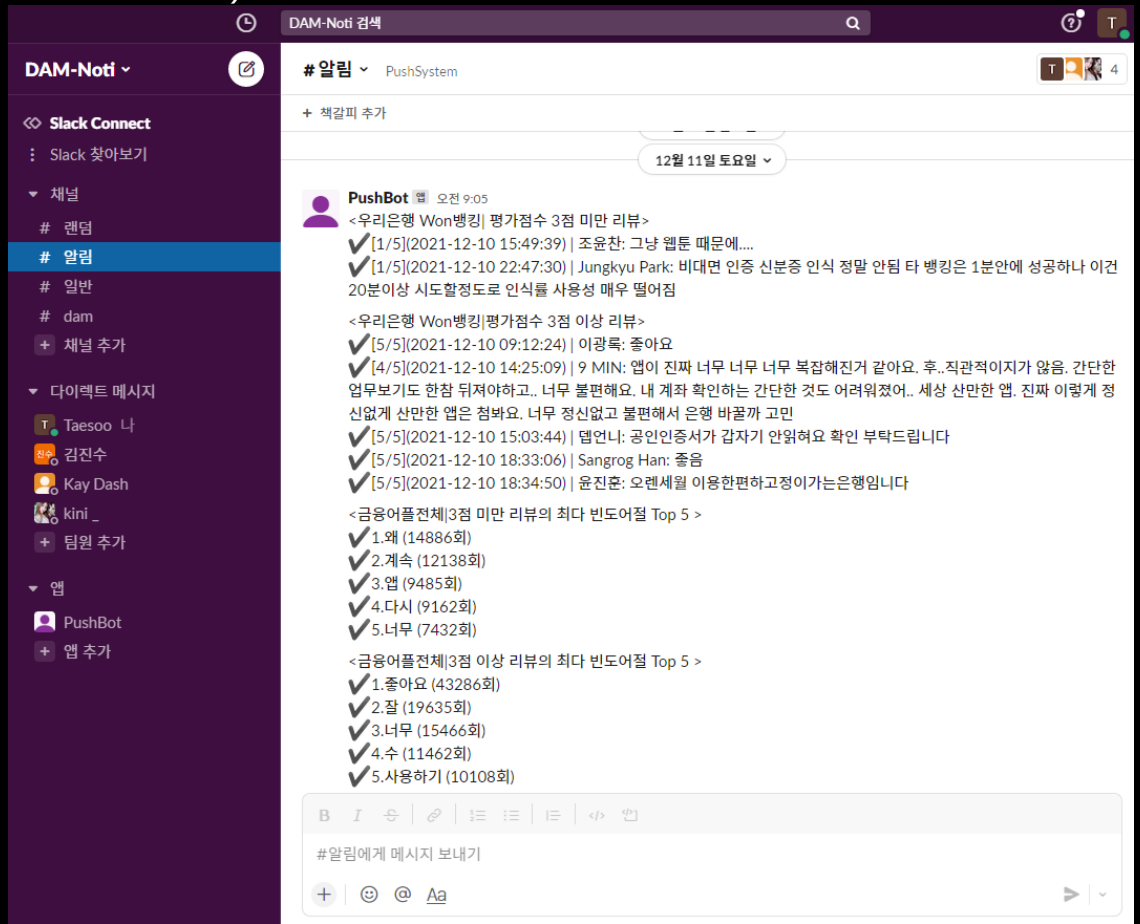
Web Service

Text Analytics



Push Service

App Push(Slack, daily AM 09:00 ~ 09:05 Push)
https://app.slack.com/



Data Lineage

Level	Task(For)	Data(Do)	Source(From)	Destination(To)	Parallel Proc(How Many)	ETL Type(How to)	ETL Method(What)	Note.(Memo)	Loop
0	데이터 스키마 정의	Business	Psuedo Code	Logical Objects	1	Abstraction to Knowledge	Realize	Create Table	Once
0	테이블 파티셔닝	Split	Table	Partitioning Table	1	-	-	for Performance	Once
0	인덱스 생성	B*Tree	RootNode	LeafNodes	1	-	-	for Performance	Once
0	모니터링 환경 구축	Performance Data	Resource Usage	WebService	1	text To Visual	Data Gathering	Prepare for DataLineage	Once
1	웹 데이터 수집	Web Documents	GooglePlayStore	ServerStorage	20	Text To File	Web Crawling	Incremental Data	Daily
2	원천 테이블 적재	ReviewTextFile	ServerStorage	DBMS	20	File To DBMS	DBMS Injection	RDBMS(MySQL)	Daily
3	데이터 클렌징	Partitioned Tables	Raw Table	Raw Table	20	-	-	Cleansing Duplicated Data	Daily
4	요약 테이블 생성	Partitioned Tables	Raw Table	Summary Table	1	-	-	Summary(Score/Perf/MainDept)	Daily
5	보고서 테이블 생성	Partitioned Tables	Summary Table	Report Table	1	-	-	Report(Score/Perf/MainDept)	Daily
6	테이블 최적화	Statistics	Raw Data	Metadata	1	-	-	27 Tables	Daily
7	WordCloud 이미지 생성	Image Files	Data	Image	1	Data to MediaFiles	DataProcessing	Renewal Imgs.	Daily
8	Text 추출	TextFile	Data	TextFile	1	Data to Text	DataProcessing	Extract Text	Daily
9	HDFS 업로드	put HDFS	TextFile	HDFS	1	put HDFS	put HDFS	TextFile to HDFS	bit15
10	WordCount	Word Count	HDFS(input)	HDFS(output)	1	WordCount	hadoop WordCount	hdfs jar	Daily
11	word Ranking	Ranking	HDFS(output)	Data	1	HDFS to Data	Store Ranking Data	Data	Daily
12	Push Msg	Push Msg	Data	Push Msg	1	Data to Push Msg	Push	Slack	Daily

Batch TimeLine

```
[root@localhost ~]# crontab -l
5  0 * * * sh /Data/Batch/1_removeCSV.sh                          > /Data/Batch/1_removeCSV.log                          2>&1      # start At. 00:05       # 1 min
10 0 * * * sh /Data/Batch/3_getNewFiles.sh                       > /Data/Batch/3_getNewFiles.log                        2>&1      # start At. 00:10       # 15 min
25 1 * * * sh /Data/Batch/4_loadAllFiles.sh                      > /Data/Batch/4_loadAllFiles.log                       2>&1      # start At. 01:25       # 5 min
30 1 * * * sh /Data/Batch/9_deleteDupRows.sh                     > /Data/Batch/9_deleteDupRows.log                      2>&1      # start At. 01:30       # 5 min(No.9 is faster after others.)
35 1 * * * sh /Data/Batch/6-2_summary.sh                         > /Data/Batch/6-2_summary.log                          2>&1      # start At. 01:35       # 10 min(parallel with 6-2,3,4)
45 1 * * * sh /Data/Batch/6-3_getPerfData.sh                     > /Data/Batch/6-3_getPerfData.log                      2>&1      # start At. 01:45       # 10 min(parallel with 6-2,3,4)
55 1 * * * sh /Data/Batch/6-4_genWordCloud.sh                    > /Data/Batch/6-4_genWordCloud.log                     2>&1      # start At. 01:55       # 60 min(parallel with 6-2,3,4)
00 2 * * * sh /Data/Batch/6_OPTIMIZE_TABLE.sh                    > /Data/Batch/6_OPTIMIZE_TABLE.log                     2>&1      # start At. 03:00       # 30 min
00 4 * * * sh /Data/Batch/8_getRows.sh                           > /Data/Batch/8_getRows.log                            2>&1      # start At. 04:00       # 1 min
30 4 * * * /IDEs/anaconda3/bin/python /Data/Batch/modelTrainer.py > /Data/Batch/modelTrainer.log                         2>&1      # start At. 04:30       # 2h 10 min
00 8 * * * /IDEs/anaconda3/bin/python /Data/Batch/B01_word_count_hadoop.py > /Data/Batch/B01_word_count_hadoop.log      2>&1      # start At. 08:00       # 30 min
00 9 * * * sh /Data/Batch/A01_enqueue_avg_msgs.sh                > /Data/Batch/A01_enqueue_avg_msgs.log                 2>&1      # start At. 09:00       # 1 min
05 9 * * * /IDEs/anaconda3/bin/python /Data/Batch/A02_dequeue_avg_msgs.py > /Data/Batch/A02_dequeue_avg_msgs.log        2>&1      # start At. 09:05       # 1 min
```

Data Interaction Process

RDBMS -> HDFS(WordCount) -> RDBMS(Rank)

Text Cleans

```
for index, row in meta.iterrows():
    sql='''
select trim(original_content) as original_content from {tab_full_nm} where score<3.0
and trim(original_content) is not null and length(trim(original_content))>0 '''.
format(tab_full_nm=row['full_name'])
    temp_data = pd.read_sql_query(sql,connection)
    temp_data['original_content'] = temp_data['original_content'].str.replace(
    "[^ㄱ-ㅎㅏ-ㅣ가-힣 1234657890]","") #remove_chars
    temp_data.to_csv('''/Data/Batch/word_count/source/low/{file_name}.csv'''.format(
    file_name=row['full_name']), index=None, header=None,encoding='utf-8-sig')
```

Put HDFS

```
os.system('su - hdroot -c "/data/hadoop-3.3.1/bin/hdfs dfs -put
/Data/Batch/word_count/source/low/*.csv  /input/low/"')
os.system('su - hdroot -c "/data/hadoop-3.3.1/bin/hdfs dfs -put
/Data/Batch/word_count/source/high/*.csv /input/high/"')
```

Ranked Data Push Process

RDBMS -> HDFS(WordCount) -> RDBMS(Rank)

Data Ranking

```
sql='''update summary.under_avg_msg set send_flag=1 where send_flag=0 and
at >= date_add(now(), interval - 72 hour)'''
cursor.execute(sql)
connection.commit()
sql='''select score,at,username,original_content,reviewid from
summary.under_avg_msg where send_flag=1 order by at asc'''
low_rvs= pd.read_sql_query(sql,connection)
sql='''update summary.under_avg_msg set send_flag=2 where send_flag=1'''
cursor.execute(sql)
connection.commit()
```

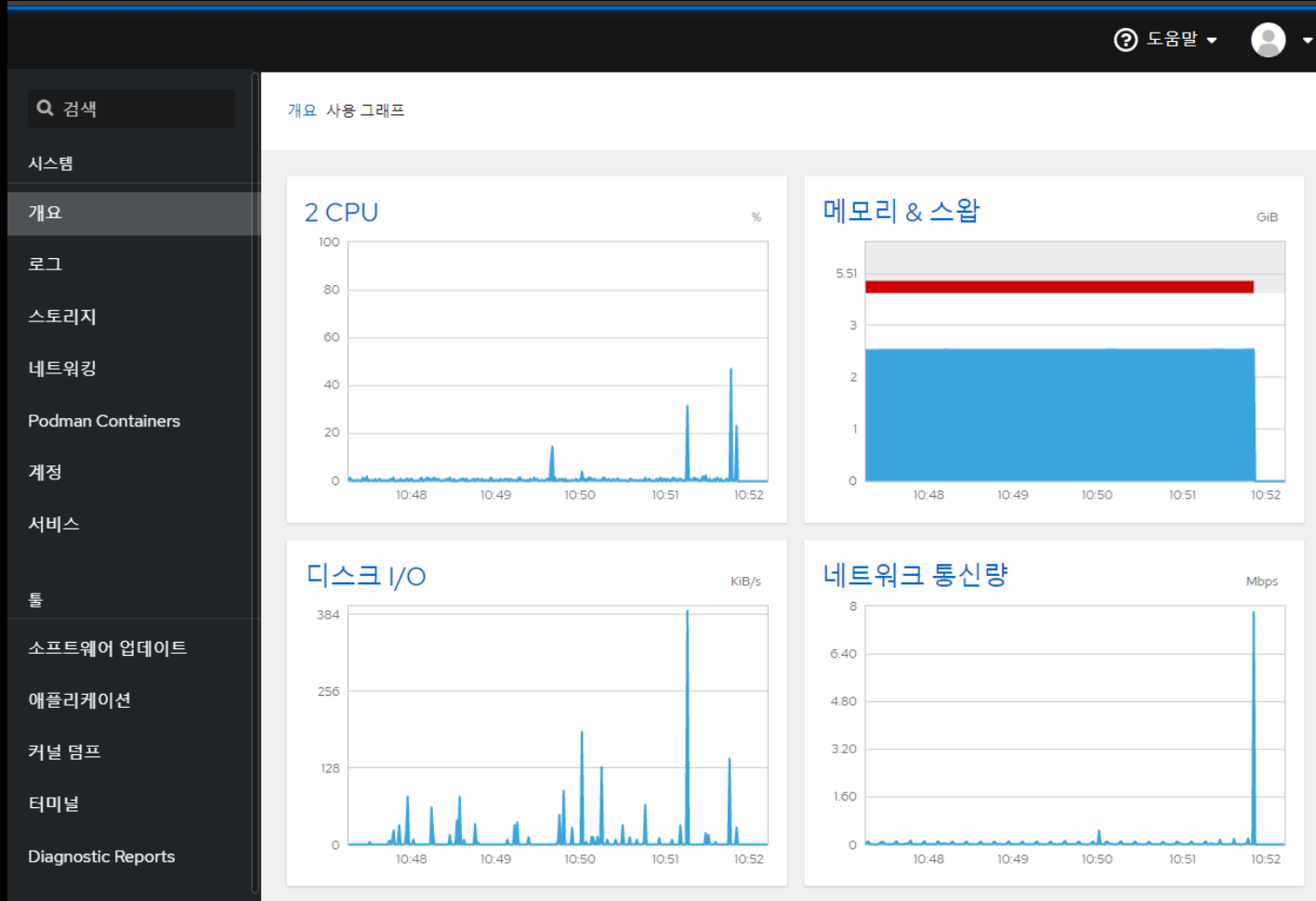
Enqueue/Dequeue Message(Push)

```
msg_txt='''<금융어플전체| 3점△ 리뷰의 최다 빈도어절 Top 5 >
'''
i=0
for index, row in high_words.iterrows():
    i=i+1
    msg_txt=msg_txt+''' ✔{ranking}.{words} ({count}회)
'''.format(ranking=int(i),words=row['word'],count=int(row['cnt']))

if len(high_words)>0:
    post_message(myToken, chnl, msg_txt)
print(msg_txt)
```

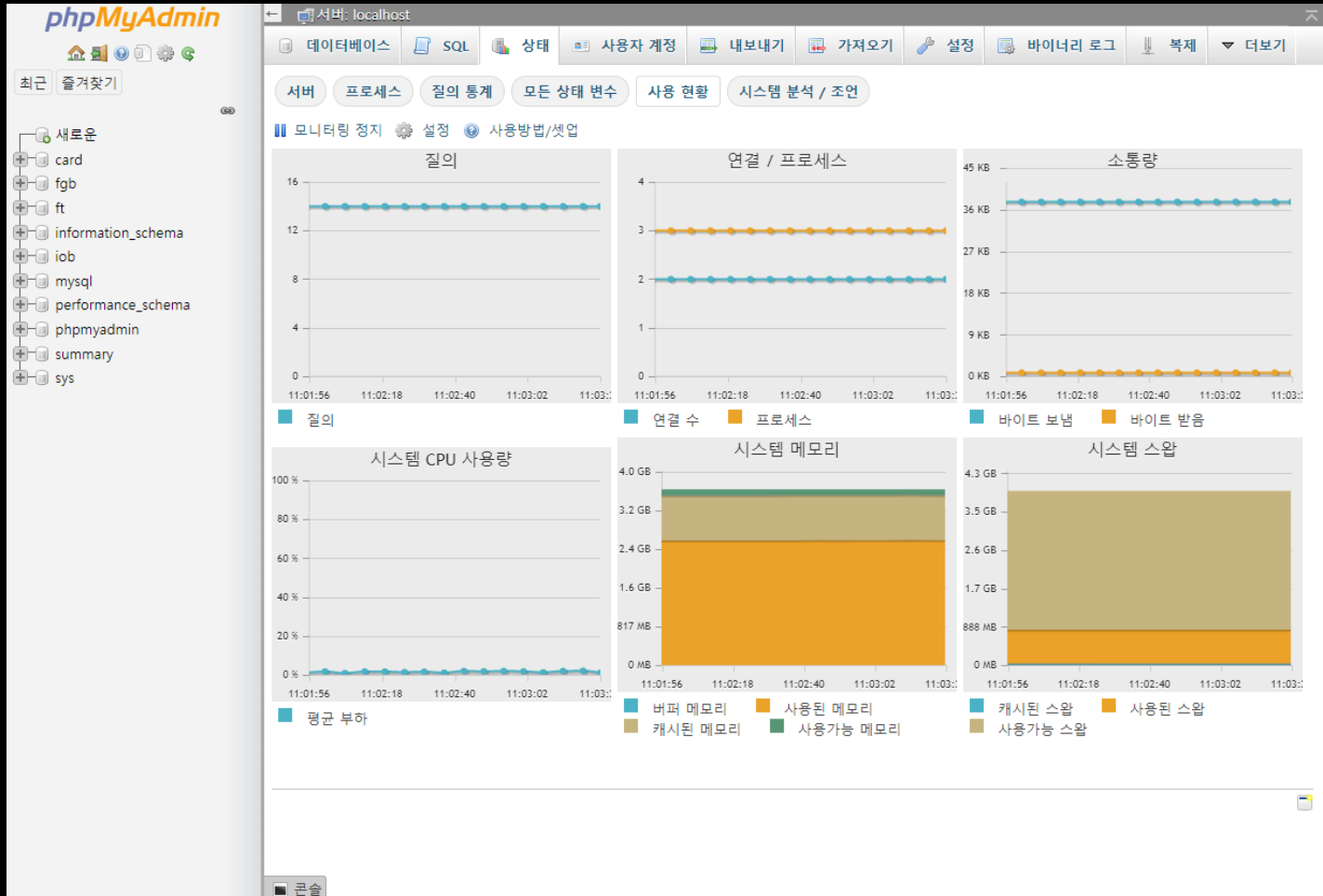
Monitoring for Operation

Server(Linux)



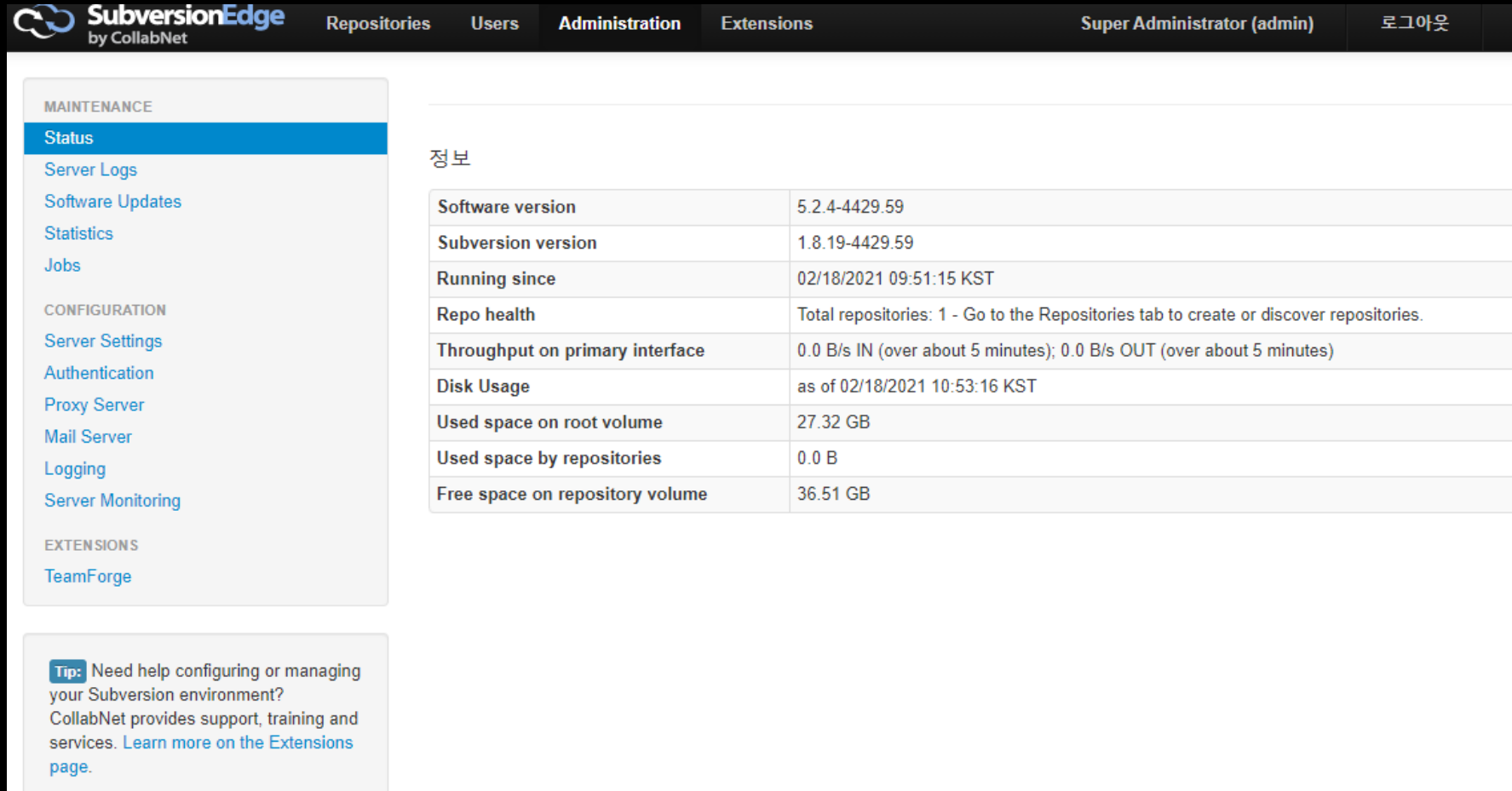
Monitoring for Operation

DBMS(MySQL)



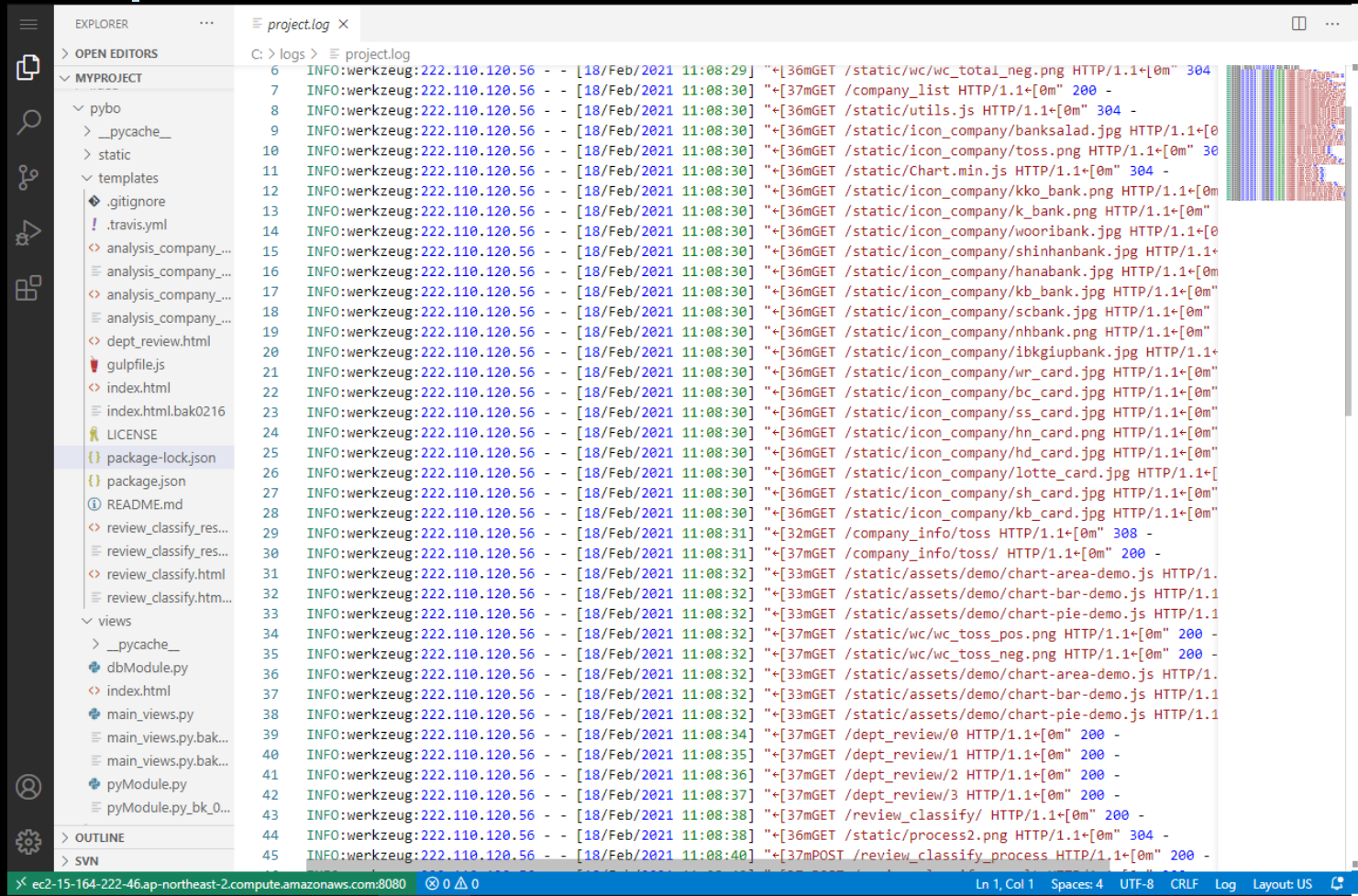
Monitoring for Operation

SVN



Monitoring for Operation

Web Server(Flask)



기대 효과

프로젝트 기대효과 및 활용방안

기대 효과

- 입력 한 금융사의 기능/비기능적 강/약점을 한눈에 통해 정보 제공
- 타 금융사의 장,단점을 분석하고 비교하여 우리금융의 개선방향 도출

활용방안

개선점 자동 알림

우리WON뱅킹 외
우리 금융 앱
개선의견 자동 알림
-> 서비스 품질 개선

AI 금융 컨설팅

마이데이터 결합
-> 고객에게 적합한
맞춤형 서비스 제공

신규 서비스 수요 예측

고객 선호 서비스 학습
-> 신사업 성공여부 예측

THANKS