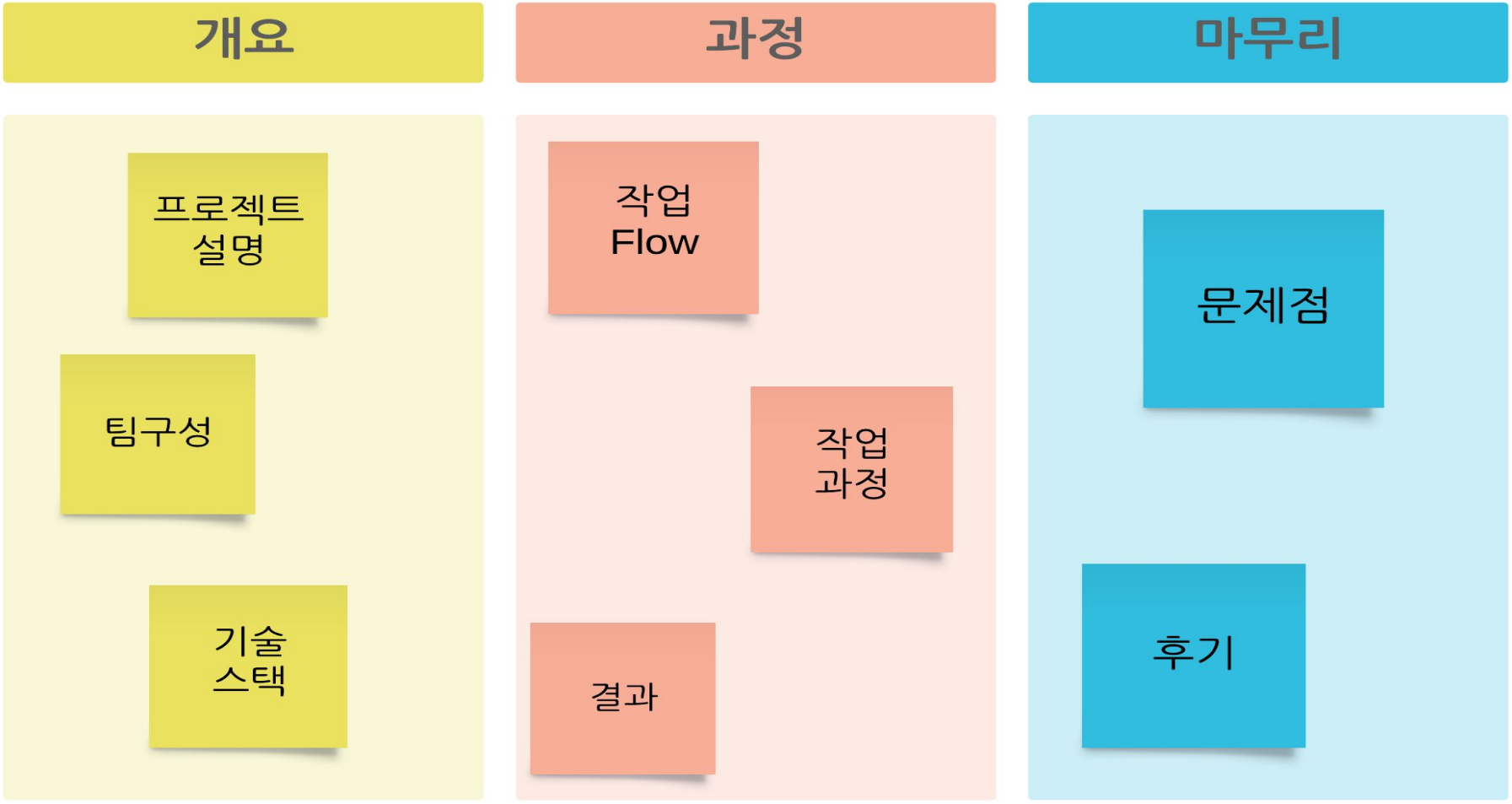
경제 기사 키워드 분석 프로젝트

Team 2 : きせき

팀원 : 기석광, 조명아, 최태성, 차민혁, 신소영

순서



프로젝트 개요

- 금융권에 관심있는 팀원들이 모여 하루동안의 경제 뉴스의 주된 토픽을 알아보기 위해 진행된 프로젝트
- 프로젝트 이전까지 배운 내용에 대한 복습 필요
- 네이버 경제 기사 크롤링 후 기사 제목과 내용을 저장
- 데이터 전처리를 통한 키워드 추출
- 추출한 데이터를 웹서버를 통해 시각화

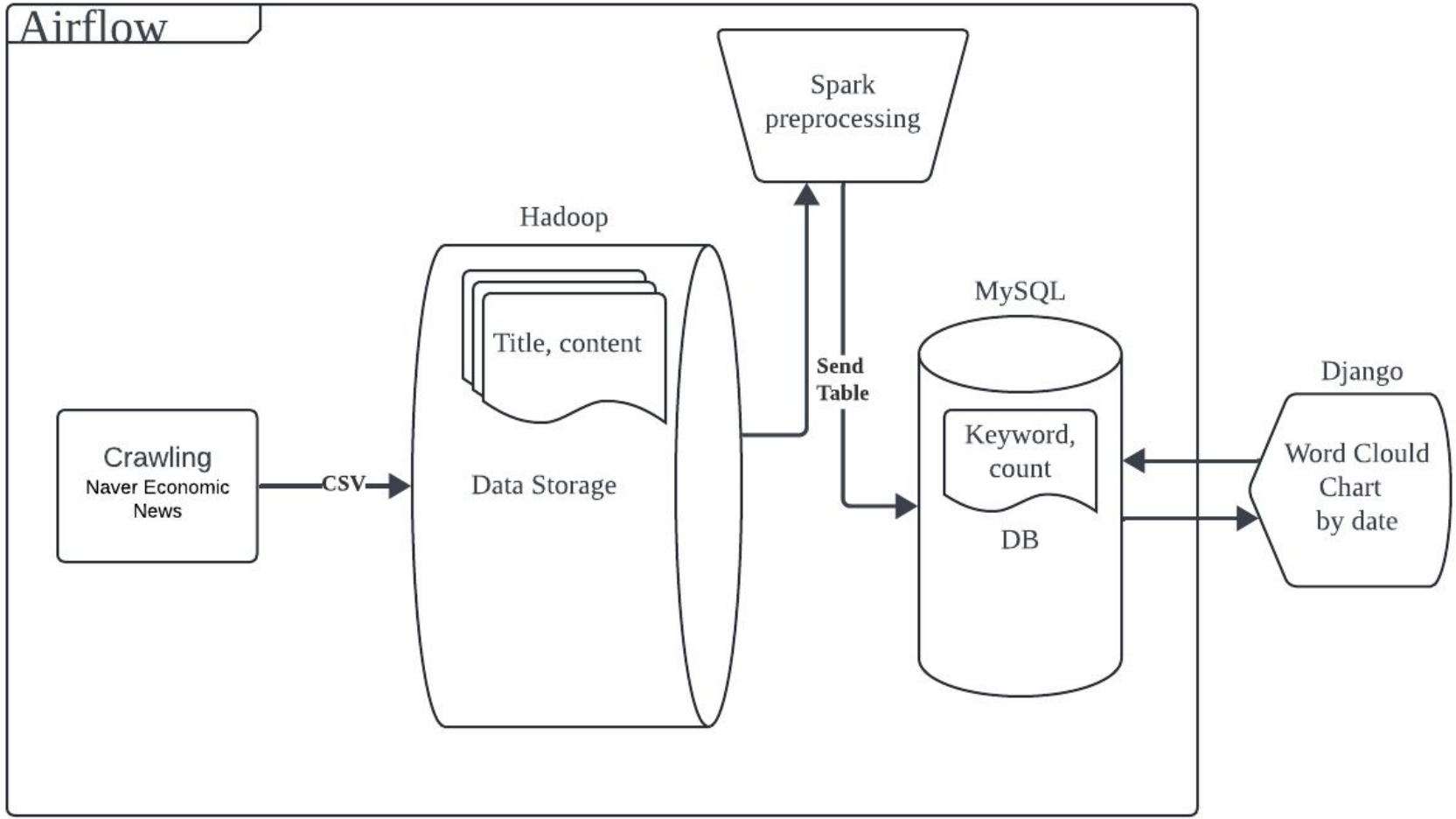
팀구성

기석광	데이터 크롤링, Airflow 구축 및 파이프라인 관리
최태성	Hadoop 구축, 키워드 추출 전처리 진행, Spark 구축 및 Airflow 보조
조명아	Spark Cluster 구축, 전처리 데이터 DB연결, 크롤링 및 전처리 보조
신소영	Django를 이용한 웹서버 구축, 홈화면과 워드클라우드 구현
차민혁	Django를 이용한 웹서버 구축. Chart.js를 활용한 차트 구현

기술 스텍

-
-
-

작업 Flow


Airflow
Spark
preprocessing
Hadoop
Title, content
Data Storage
MySQL
Keyword,
count
DB
Send
Table
Django
Word Clould
Chart
by date
Crawling
Naver Economic
News
CSV

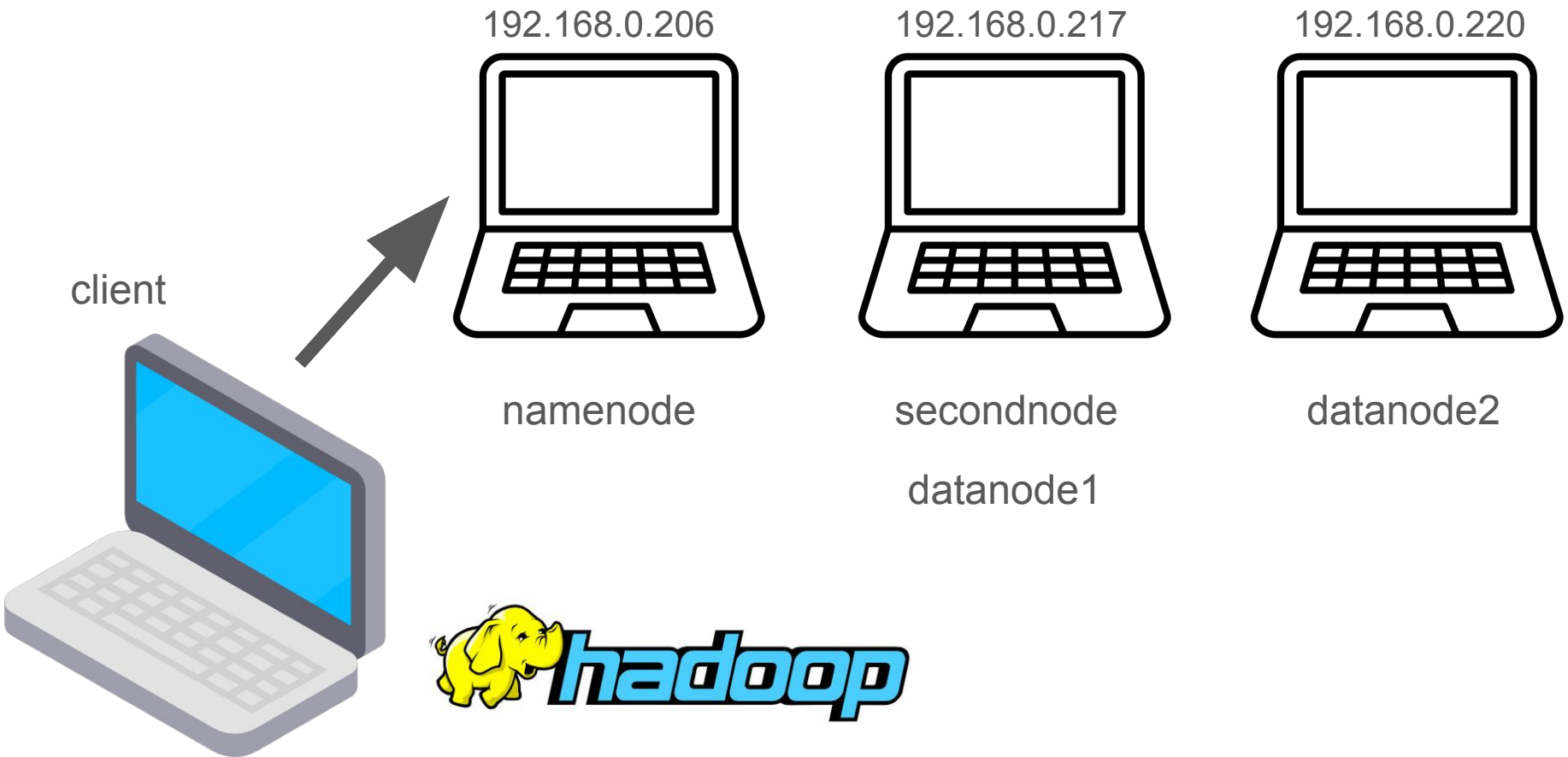
네이버 뉴스 크롤링

asyncio는 async/await 구문을 사용하여 **동시성** 코드를 작성하는 라이브러리
aiohttp는 비동기 GET요청을 위한 라이브러리

```python
async def get_news_content(session, url):
    response_text = await fetch(session, url)
    if response_text is None:
        return None, None
    soup = BeautifulSoup(response_text, 'html.parser')

    title_tag = soup.select_one("h2.media_end_head_headline")
    content_tag = soup.find('article', {'id': 'dic_area'})

    if title_tag and content_tag:
        title = title_tag.get_text().strip()
        content = content_tag.get_text().strip()
        return title, content
    return None, None
```


```python
tasks = [get_news_content(session, link) for link in news_links]

# news_contents = await asyncio.gather(*tasks)
news_contents = await tqdm_asyncio.gather(*tasks, desc="Fetching news content")
```

하둡(Hadoop) 구축


192.168.0.206
192.168.0.217
192.168.0.220
client
namenode
secondnode
datanode1
datanode2
hadoop

하둡(Hadoop) 구축

Overview 'namenode:8020' (✔active)

Started:	Tue Jul 02 15:29:29 +0900 2024
Version:	3.3.6, r1be78238728da9266a4f88195058f08fd012bf9c
Compiled:	Sun Jun 18 17:22:00 +0900 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
Cluster ID:	CID-aaed7d42-2831-474a-ac95-e93bd433d88b
Block Pool ID:	BP-1115767673-127.0.1.1-1719898770461

Summary

Security is off.

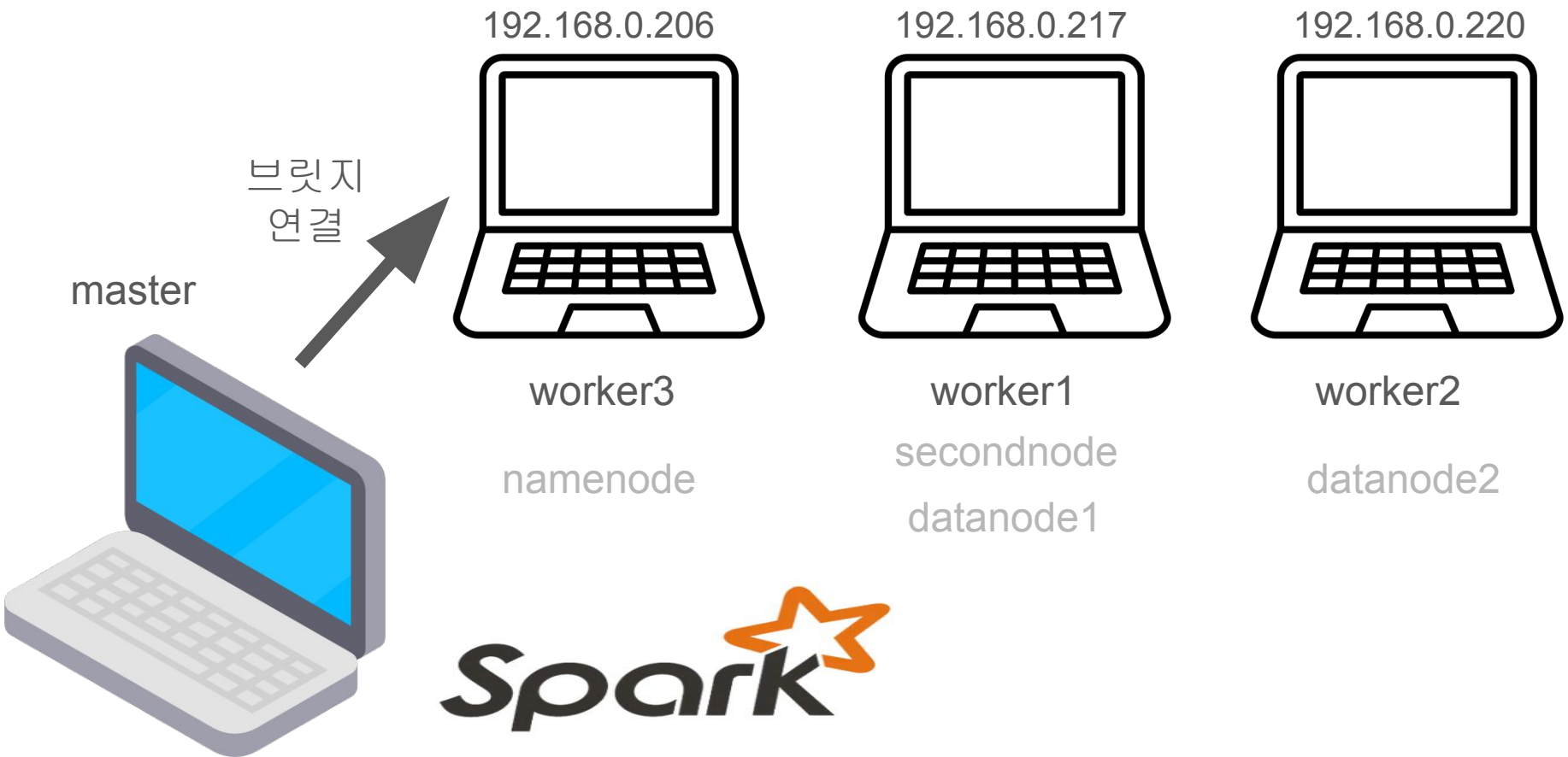
Safemode is off.

34 files and directories, 14 blocks (14 replicated blocks, 0 erasure coded block groups) = 48 total filesystem object(s).

Heap Memory used 110.63 MB of 391 MB Heap Memory. Max Heap Memory is 3.88 GB.

Non Heap Memory used 93.14 MB of 96.19 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

스파크(Spark) 구축


192.168.0.206
192.168.0.217
192.168.0.220
브릿지
연결
master
worker3
namenode
worker1
secondnode
datanode1
worker2
datanode2
Spark

스파크(Spark) 구축

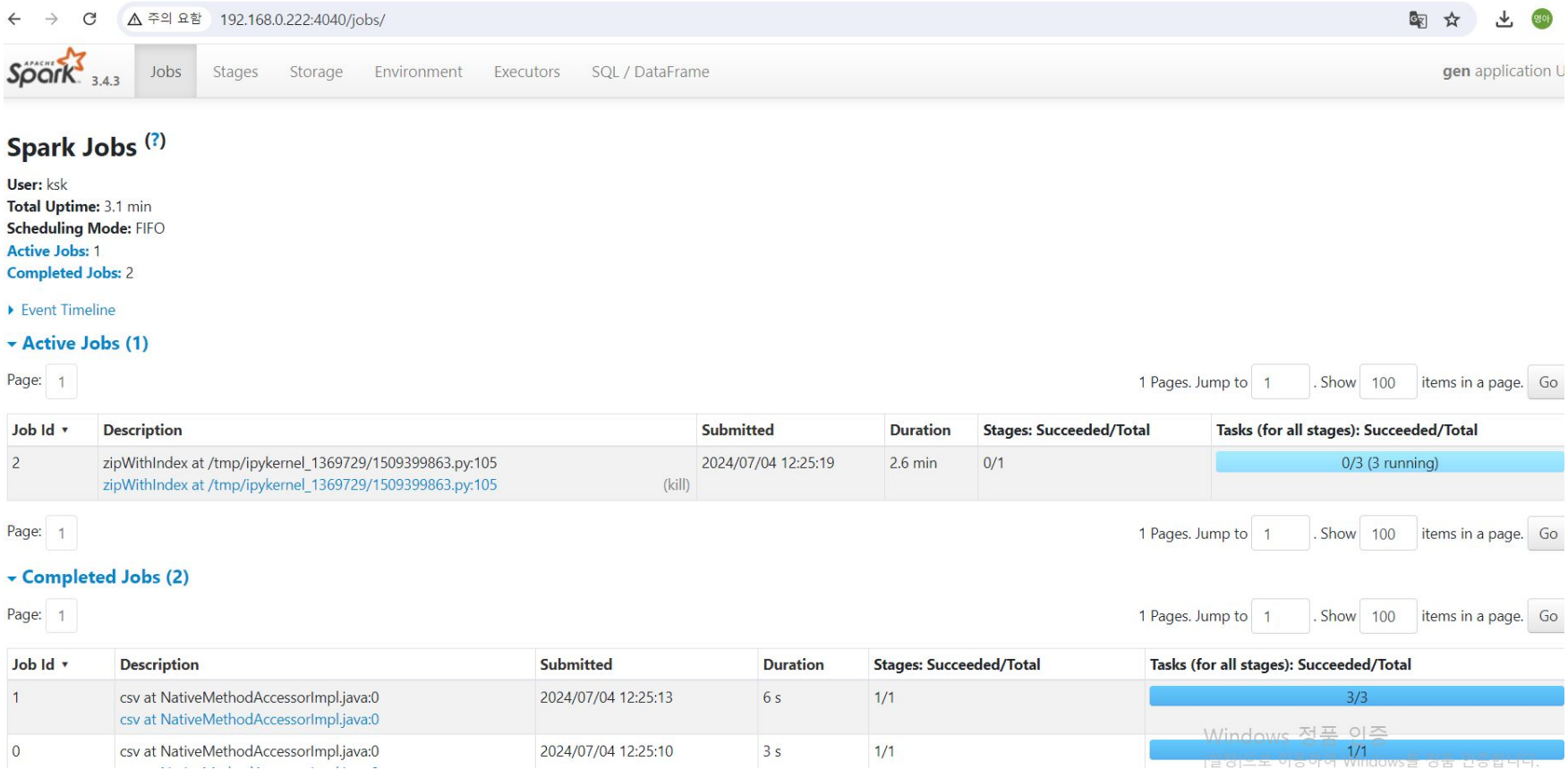
spark-env.sh

```
export SPARK_MASTER_HOST=master
export SPARK_MASTER_PORT=7077
export SPARK_MASTER_WEBUI=8090
export SPARK_WORKER_CORES=4
export SPARK_WORKER_MEMORY=4g
export SPARK_WORKER_INSTANCES=3
```

spark-defaults.conf

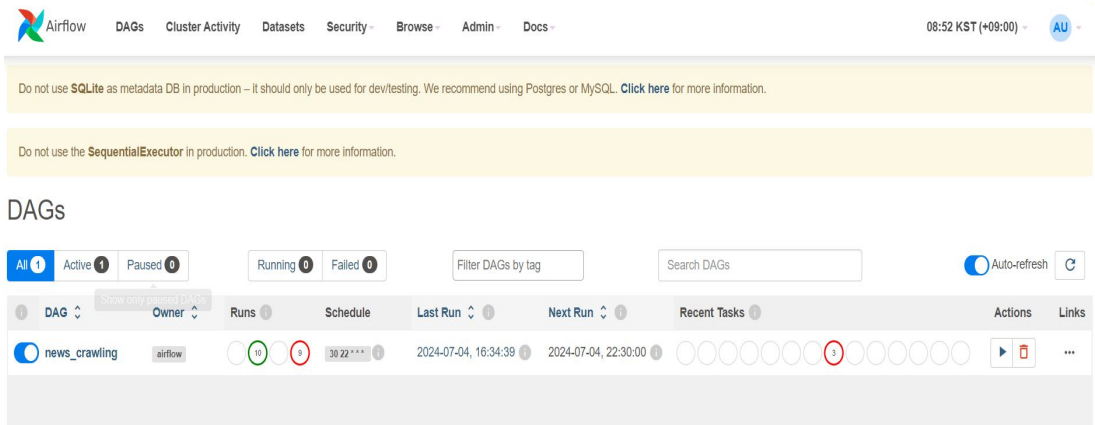
```
spark.executor.cores=4
spark.executor.memory=4g
spark.sql.shuffle.partitions=400
spark.dynamicAllocation.enabled=true
spark.dynamicAllocation.minExecutors=2
spark.dynamicAllocation.maxExecutors=10
```

스파크(Spark) 구축


주의 요함
192.168.0.222:4040/jobs/
Spark 3.4.3
Jobs
Stages
Storage
Environment
Executors
SQL / DataFrame
gen application U
Spark Jobs (?)
User: ksk
Total Uptime: 3.1 min
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 2
Event Timeline
Active Jobs (1)
Page: 1
1 Pages. Jump to 1 . Show 100 items in a page. Go
Job Id
Description
Submitted
Duration
Stages: Succeeded/Total
Tasks (for all stages): Succeeded/Total
2
zipWithIndex at /tmp/ipykernel_1369729/1509399863.py:105
zipWithIndex at /tmp/ipykernel_1369729/1509399863.py:105 (kill)
2024/07/04 12:25:19
2.6 min
0/1
0/3 (3 running)
Page: 1
1 Pages. Jump to 1 . Show 100 items in a page. Go
Completed Jobs (2)
Page: 1
1 Pages. Jump to 1 . Show 100 items in a page. Go
Job Id
Description
Submitted
Duration
Stages: Succeeded/Total
Tasks (for all stages): Succeeded/Total
1
csv at NativeMethodAccessorImpl.java:0
csv at NativeMethodAccessorImpl.java:0
2024/07/04 12:25:13
6 s
1/1
3/3
0
csv at NativeMethodAccessorImpl.java:0
2024/07/04 12:25:10
3 s
1/1
1/1

Airflow 구축

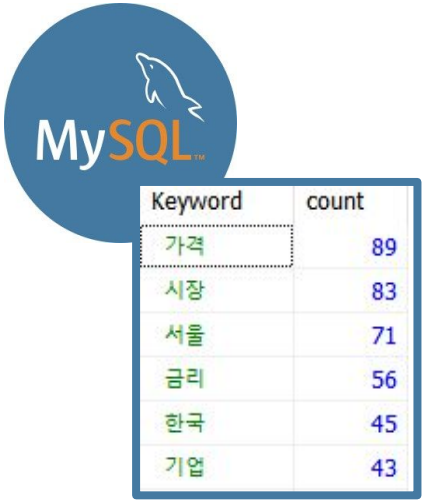
```
#
dags_folder = /home/ksk/project/third_project/dags
```




```
sys.path.append(os.path.join(os.path.dirname(__file__), '../modules'))

# news_crawling 모듈 임포트
from news_crawling import scrape_news, transfer_to_hdfs
# from keyword_ext
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2024, 6, 30),
    'retries': 3,
    'retry_delay': timedelta(minutes=5),
}

dag = DAG(
    dag_id='news_crawling',
    description="Daily news crawling",
    default_args=default_args,
    schedule_interval='30 22 * * *',  # 매일 밤 23:00에 실행
    catchup=False,
```

키워드 추출 & count

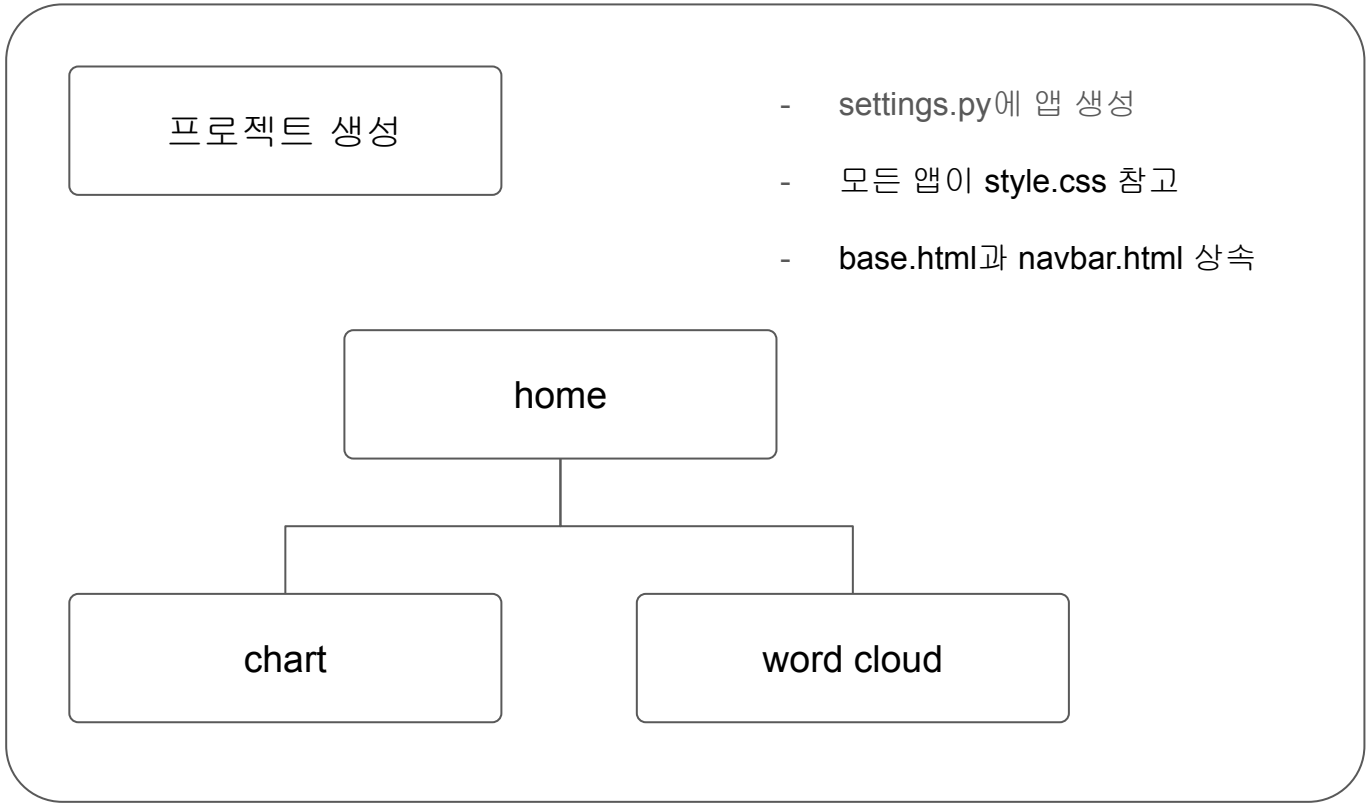
Block Size	Name
128 MB	2024-06-26.csv
128 MB	2024-06-28.csv
128 MB	2024-06-29.csv
128 MB	2024-06-30.csv
128 MB	2024-07-01.csv
128 MB	2024-07-02.csv
128 MB	2024-07-03.csv
128 MB	2024-07-04.csv



1. Hadoop에 저장된 뉴스 기사 csv 파일을 스파크에서 불러옵니다.
2. bareunpy 모듈을 사용하여 기본 키워드를 추출합니다.
3. scikit-learn의 cosine similarity로 최대 마진 적중률(MMR)을 계산하여 키워드를 추출합니다.
4. 추출된 키워드를 단어 별로 카운트 한 후 해당 데이터를 테이블로 변환하여 DB에 저장합니다.
5. 테이블명은 각 날짜로 지정하여 Django에서 각 날짜에 맞게 데이터를 불러올 수 있도록 합니다.

Django 프로젝트 생성

```
myproject/
├── web/
│   ├── chart/
│   │   ├── models.py
│   │   ├── urls.py
│   │   └── views.py
│   ├── home/
│   │   ├── models.py
│   │   ├── urls.py
│   │   └── views.py
│   └── word/
│       ├── models.py
│       ├── urls.py
│       └── views.py
├── config/
│   ├── settings.py
│   └── urls.py
├── static/
│   ├── fonts/
│   └── style.css
└── templates/
    ├── chart/
    │   ├── index.html
    │   └── chart.html
    ├── base.html
    └── navbar.html
```



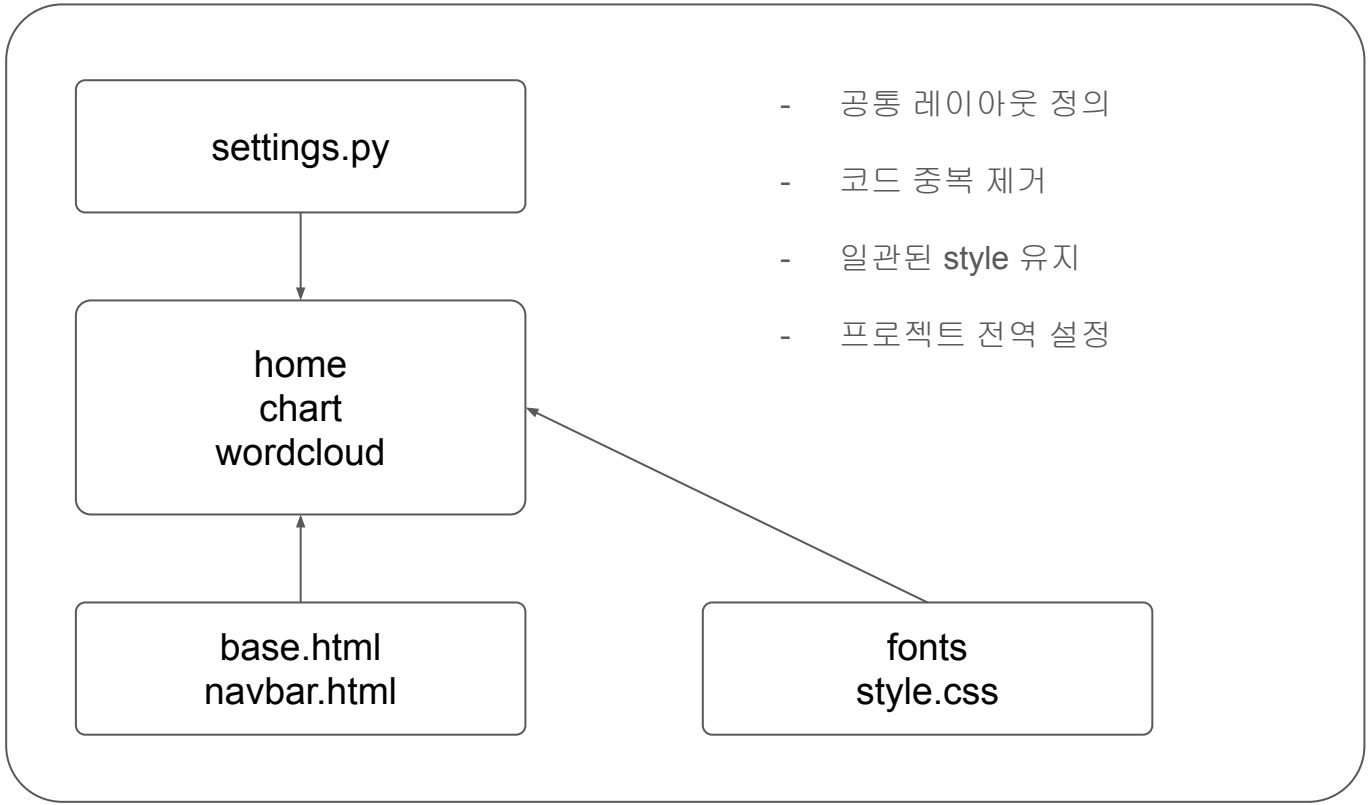
Django 모델, 뷰, 템플릿 작성

```
myproject/
├── web/
│   ├── chart/
│   │   ├── models.py
│   │   ├── urls.py
│   │   └── views.py
│   ├── home/
│   │   ├── models.py
│   │   ├── urls.py
│   │   └── views.py
│   └── word/
│       ├── models.py
│       ├── urls.py
│       └── views.py
├── config/
│   ├── settings.py
│   └── urls.py
├── static/
│   ├── fonts/
│   └── style.css
├── templates/
│   ├── chart/
│   │   ├── index.html
│   │   └── chart.html
│   ├── base.html
│   └── navbar.html
```

home	chart	word
• DB 에서 데이터 조회 • 전일 최대 Keyword 표시 • 버튼 링크화 • chart, word 앱으로 이동	• DB에 동적 접근 • Chart.js 이용 • 파이 차트와 바 차트 • top keywords 리스트	• 입력 받은 input값으로 mysql 테이블 조회 • 조회 결과를 바탕으로 워드클라우드 이미지제공

Django 설정 파일과 정적 파일 수정

```
myproject/
├── web/
│   ├── chart/
│   │   ├── models.py
│   │   ├── urls.py
│   │   └── views.py
│   ├── home/
│   │   ├── models.py
│   │   ├── urls.py
│   │   └── views.py
│   └── word/
│       ├── models.py
│       ├── urls.py
│       └── views.py
├── config/
│   ├── settings.py
│   └── urls.py
├── static/
│   ├── fonts/
│   └── style.css
├── templates/
│   ├── chart/
│   │   ├── index.html
│   │   └── chart.html
│   ├── base.html
│   └── navbar.html
```



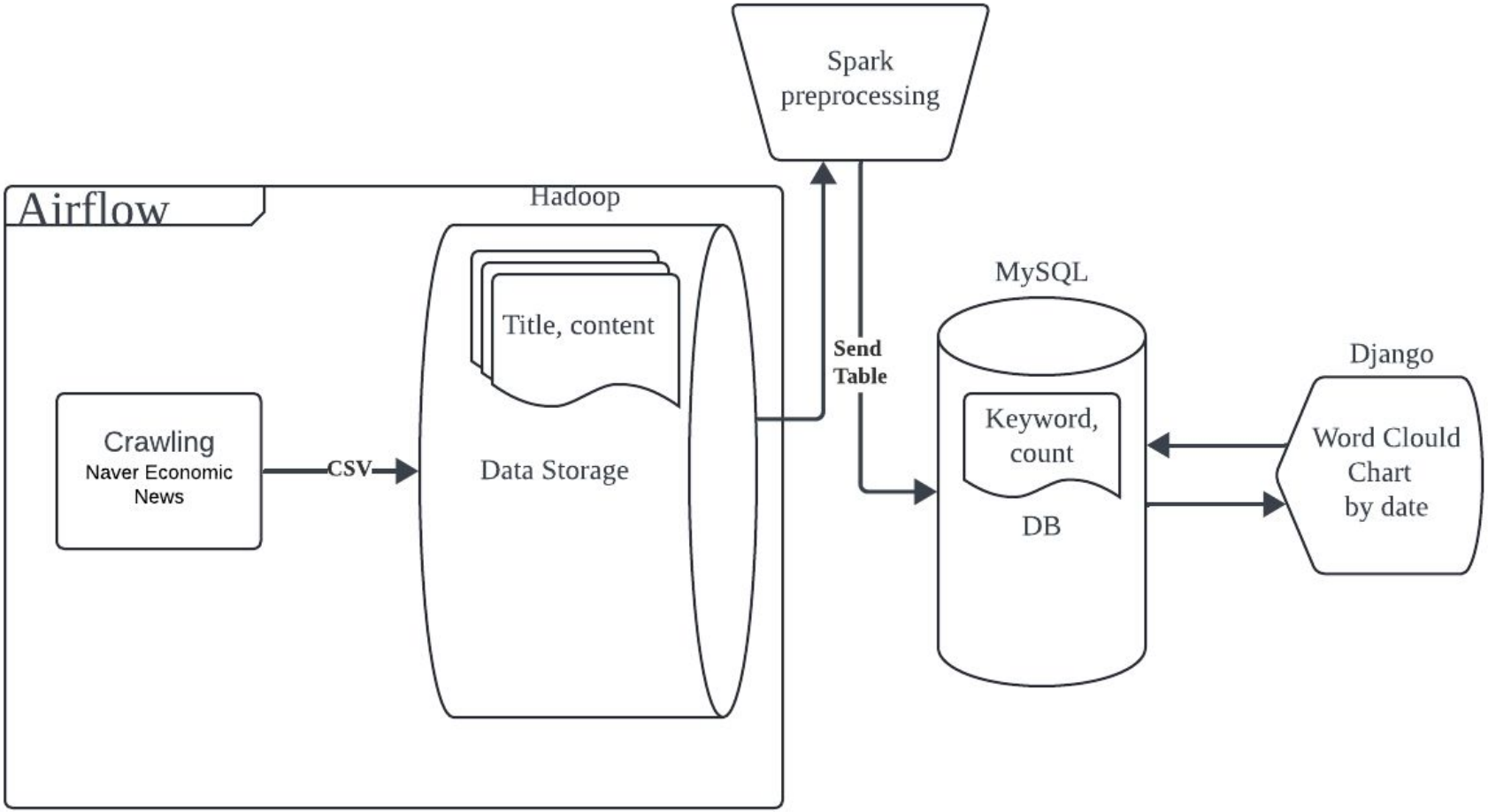
문제점


protobuf
Protocol Buffers

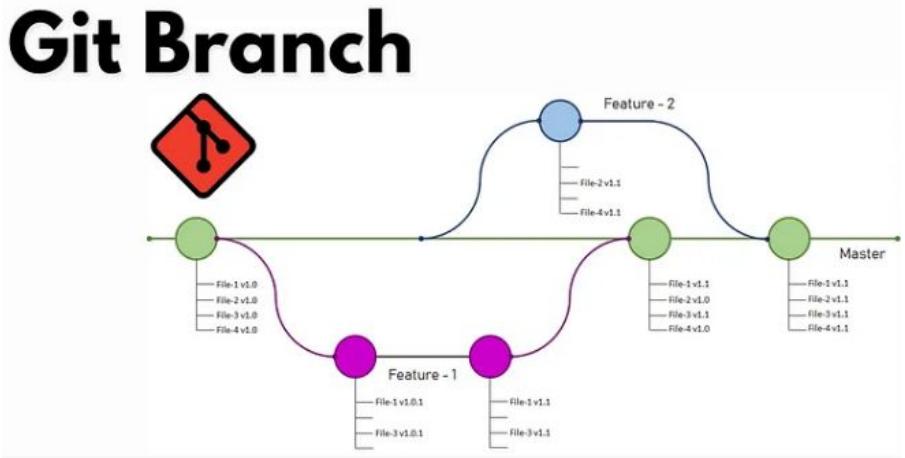

bareun


Apache
Airflow
APACHE
Spark

진행된 Flow


Spark
preprocessing
Airflow
Hadoop
Title, content
Data Storage
Crawling
Naver Economic
News
CSV
Send
Table
MySQL
Keyword,
count
DB
Django
Word Clould
Chart
by date

후기



공통된 후기

- 역할 분담이 잘 되었다.
- 수업에서 배웠던 내용을 복습할 수 있어서 좋았다.
- 프로젝트를 진행하며 Git을 잘 활용하지 못 한 게 아쉽다.
- Hadoop, Airflow, Spark 파트와 Django 파트의 소통이 부족했던 것 같아 아쉽다.