최종 포팅 메뉴얼

1. 인프라

1. 개발 환경

- **Server** : Ubuntu 20.04 LTS
- **FrontEnd**
    - **Node.js** : 18.13.0
    - **React:** 18.2.0
- **Spring Server**
    - **JDK**: OpenJDK11
    - **SpringBoot**: 2.7.9

- **Django Server**
  - **Django** : 4.2
  - **Python** : 3.10.10
- **FastAPI Server**
  - **FastAPI** : 0.45
  - **Python**: 3.7
- **Infra**
  - **Nginx** : 1.23.3
  - **Jenkins** : 2.375.2
- **Tools**
  - **Vscode**: : 1.73.1
  - **IntelliJ** : 2022.03
- **Database**
  - **MySQL** : 8.0.32
  - **Redis** : 7.0.8

2. 설정파일 및 환경 변수 정보

▼ Spring

▼ applicationl.yml

```
server:
  port: 8080
  servlet:
    context-path: /api

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    urL : jdbc:mysql://j8a508.p.ssafy.io/stockey?serverTimezone=Asia/Seoul
    username: develop
    password: develop



  jpa:
    open-in-view: true
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate.format_sql: true

logging:
  level:
    org.hibernate.SQL: debug


# springdoc swagger
springdoc:
  api-docs:
    path: /docs # 접속 path 설정
  swagger-ui:
    path: /swagger-ui # 접속 path 설정

kakaoOauth:
  REST_API_KEY: 204f458585e0229e8443cd7bc1be5c5e
  REDIRECT_URL: http://localhost:3000/oauth/kakao
```

```
jwt:
  # base64로 인코딩된 암호 키, HS512를 사용할 것이기 때문에, 512비트(64바이트) 이상이 되어야 합니다. 길게 써주세요
  secretKey: c3NhZnk46riwMu2Vmeq4sOqzte2Gte2UhOuhnOygne2KuEEzMDg=
  access:
    expiration: 100
  refresh:
    expiration: 14400


django:
  url: http://j8a508.p.ssafy.io
  port : 8082
```

▼ 환경 변수 .env 파일

```
PORT=8080
DATABASE_URL=jdbc:mysql:<URL>:<PORT>/<DB명>?serverTimezone=Asia/Seoul
DATABASE_USERNAME=<DB명>
DATABASE_PASSWORD=<비밀번호>
KAKAO_KEY=<카카오키>
KAKAO_REDIRECT_KEY=<카카오 리다이렉트>
JWT_SECRET_KEY=<JWT 비밀번호>
DJANGO_PORT=<DJango 포트>
```

▼ React

환경변수 파일 .env.production

```


REACT_APP_KAKAO_API = <카카오 API>
REACT_APP_KAKAO_REDIRECT = <카카오 Redirect>
REACT_APP_SERVER_BASE_URL =<서버 주소>
```

▼ Django

```

{
  "SECRET_KEY" : "django 시크릿 키",
  "DATABASE": {
    "USER": "유저",
    "NAME": "db명",
    "PASSWORD": "비밀번호",
    "HOST": "url",
    "PORT": 3306
  }
}
```

3. 호스트 서버에 설치한 라이브러리

▼ `Docker`

```
apt-get update
apt-get install docker.io
ln -sf /usr/bin/docker.io /usr/local/bin/docker
docker -v
systemctl start docker
sudo usermod -aG docker ubuntu
sudo chmod 666 /var/run/docker.sock
apt-get update
```

▶️ **apt 업데이트** `$ apt-get update`

▶️ **도커 설치** `$ apt-get install docker.io`

▶️ **링크 생성** `$ ln -sf /usr/bin/docker.io /usr/local/bin/docker`

▶️ **도커 버젼 확인** `$ docker -v`

▶️ **도커 데몬 실행** `$ systemctl start docker`

Sudo 명령어 없이 치기

`sudo usermod -aG docker ubuntu`

docker 모드

`sudo chmod 666 /var/run/docker.sock`

▼ **`docker-compose`**

설치

```
sudo curl -L https://github.com/docker/compose/releases/latest/\
download/docker-compose-$(uname -s sudo chmod +x usr/local/bin/docker-compose
```

설치 확인

`$docker-compose -v`

docker-compose.yml에 작성된 모든 컨테이너 생성, 실행

`$docker-compose up`

docker-compose.yml에 작성된 모든 컨테이너 중지, 종료

`$docker-compose down`

▼ **ifconfig 를 위한 `net-tools` 설치**

```
sudo apt-get install net-tools
```

▼ **`openJDK 11` 설치**

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install openjdk-11-jdk
java -version
vim ~/.bashrc

# .bashrc에 아래 두줄 추가
export JAVA_HOME=$(dirname $(dirname $(readlink -f $(which java))))
export PATH=$PATH:$JAVA_HOME/bin

source ~/.bashrc
echo $JAVA_HOME
```

4. 컨테이너간 통신을 위한 도커 네트워크 설정

▼ 컨테이너 간 통신을 위한 도커 네트워크 설정

- 도커 네트워크 생성

```
docker network create ssafy-net
```

- 도커 네트워크 연결

```
docker run -d --name <container name> --network ssafy-net <image name>
```

- 같은 네트워크 안에 있으면 컨테이너 이름으로 통신가능

5. 컨테이너

▼ mysql container

- **mysql:8.0.32 (https://hub.docker.com/_/mysql)**
- **oraclelinux:8-slim**
- ▼ 컨테이너 생성

1. **mysql 이미지 가져오기**
    a. docker hub에서 이미지 다운로드

    ```
    docker pull mysql:8.0.32
    ```

    b. 다운로드 한 이미지 확인

    ```
    docker images
    ```

2. **mysql 이미지를 이용해 컨테이너 생성**
    a. docker 컨테이너 생성 및 시작

    ```
    # docker run [option] [image] [command] [args]
    docker run (옵션) mysql:8.0.32
    ```

    ```
    docker run --name mysql-container --network ssafy-net -d -p 3306:3306 -v /home/ubuntu/mysql:/var/lib/mysql -e MYSQL_ROOT_PA
    ```

    b. (옵션) 컨테이너 이름 설정

    ```
    # --name [NAME]
    --name mysql-container
    ```

    c. (옵션) 백그라운드에서 동작

    ```
    -d
    ```

    d. (옵션) 도커 컨테이너 ip:port를 호스트의 ip:port와 연결

    ```
    #-p [HOST IP:PORT]:[CONTAINER PORT] [CONTAINER NAME]
    -p 3306:3306
    ```

    e. (옵션) 컨테이너의 디렉토리와 호스트의 디렉토리를 마운트

    ```
    # -v [호스트의 공유 디렉토리]:[컨테이너의 공유 디렉토리]
    -v /home/ubuntu/mysql:/var/lib/mysql
    ```

    f. (옵션) 컨테이너 환경변수를 설정

```
-e MYSQL_ROOT_PASSWORD=ssafy
-e TZ='Asia/Seoul'
```

▼ mysql 사용자 설정

3. 사용자 설정
    a. 현재 실행중인 컨테이너 리스트 조회

    ```
    docker ps
    docker instpect [CONTAINER NAME]
    ```

    b. 컨테이너 접속

    ```
    docker exec -it mysql-container /bin/bash
    ```

    c. mysql 쉘 접속

    ```
    mysql -uroot -p
    비번 : ssafy
    ```

    d. mysql 사용자 설정

    ```
    show databases;
    use mysql;
    select user,host from user;
    CREATE USER 'develop'@'%' IDENTIFIED BY 'develop';
    GRANT ALL ON *.* TO 'develop'@'%';
    SHOW GRANTS FOR 'develop'@'%';
    ```

▼ spring container

▼ 스프링 코드 빌드

```
chmod +x ./gradlew
./gradlew clean bootJar
```

▼ 컨테이너 생성 및 실행

```
FROM openjdk:11
ARG JAR_FILE=./build/libs/backend-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar

EXPOSE 8080
ENTRYPOINT ["java","-Dspring.profiles.active=prod", "-jar", "/app.jar"]
```

```
PORT=8080
DATABASE_URL=jdbc:mysql://j8a508.p.ssafy.io:3306/stockey?serverTimezone=Asia/Seoul
DATABASE_USERNAME=develop
DATABASE_PASSWORD=develop
KAKAO_KEY=204f458585e0229e8443cd7bc1be5c5e
KAKAO_REDIRECT_KEY=http://j8a508.p.ssafy.io:8080/oauth/kakao
JWT_SECRET_KEY=c3NhZnk46riwMu2Vmeq4sOqzte2Gte2UhOuhnOygne2KuEEzMDg=
```

이미지 생성

```
docker build --force-rm -t spring-server:1.0.0 .
```

컨테이너 실행

```
docker run -d -p 8080:8080 --name spring-container --network ssafy-net --env-file /.env spring-server:1.0.0
docker run -d -p 8080:8080 --name spring-container --network ssafy-net -v /home/ubuntu/spring/.env:/.env --env-file /.env spring-se
```

▼ Redis Container

```
docker pull redis
docker run --name redis-container --network ssafy-net -p 6379:6379 -d redis --requirepass ssafy
```

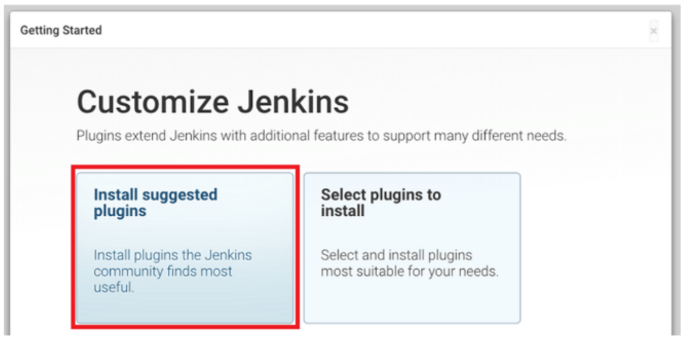
▼ jenkins container

▼ 젠킨스 컨테이너 생성 및 플러그인 설치

- 컨테이너 생성 및 실행

```
docker pull jenkins/jenkins:lts
docker run -d --name jenkins-container --network ssafy-net -p 8081:8080 -v /home/ubuntu/spring/.env:/.env -v /jenkins:/var/jenk
docker logs jenkins-container
```

- url:8081로 접속한 후 플러그인 설치


Getting Started

Customize Jenkins

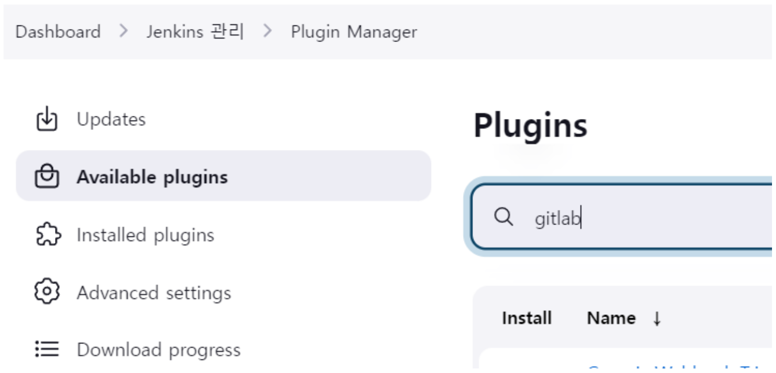
Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.

- 계정생성

```
계정명 : <계정명>
암호 : <비밀번호>
이름 : <이름>
```

- gitlab 플러그인 설치



```
gitlab
gitlab api
```

▼ **nginx container**

```
docker pull nginx:latest
docker run -d --name nginx-container --network ssafy-net -p 80:80 -p 443:443 --restart=unless-stopped -v /home/ubuntu/nginx/conf.d:/etc/
```

```
docker exec -it nginx-container /bin/bash
```

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}
```

```
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

    location / {
        proxy_pass http://react-container:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api/ {
        proxy_pass http://spring-container:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }


}
```

```
server {
    listen       80;
    listen  [::]:80;
    server_name  j8a508.p.ssafy.io;


    location /.well-known/acme-challenge {
        allow all;
        root /var/www/certbot;
    }

    location / {
       return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name j8a508.p.ssafy.io;


    ssl_certificate     /etc/nginx/ssl/live/j8a508.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/nginx/ssl/live/j8a508.p.ssafy.io/privkey.pem;

    location / {
        proxy_pass http://react-container:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api/ {
        proxy_pass http://spring-container:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

▼ react container

```
FROM node:latest

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build

# Use Nginx as the web server
FROM nginx
```

```
# nginx 의 default.conf 를 삭제
RUN rm /etc/nginx/conf.d/default.conf

# host pc 의 nginx.conf 를 아래 경로에 복사
COPY ./default.conf /etc/nginx/conf.d/default.conf

# Copy the build files to the Nginx web root directory
COPY --from=0 /app/build /usr/share/nginx/html

# Expose port 3000
EXPOSE 3000

# Start Nginx
CMD ["nginx", "-g", "daemon off;"]
```

```
server {
    listen 3000;
    root /usr/share/nginx/html;
    index index.html;

    location / {
        try_files $uri /index.html;
    }

    error_log /var/log/nginx/api_error.log;
    access_log /var/log/nginx/api_access.log;
   }
```

```
docker build --force-rm -t react-server .
```

```
docker run --network ssafy-net --name react-container -p 3000:3000 -d react-server
```

▼ django container

DockerFile

```
# Use an official Python runtime as the base image
FROM python:3.10.10
ENV PYTHONUNBUFFERED = 1

RUN apt-get -y update
RUN mkdir /app

# Set the working directory to /app
WORKDIR /app

COPY requirements.txt /app/

# Install required packages
RUN pip install --upgrade pip
RUN pip install -r requirements.txt

# Copy the current directory contents into the container at /app
COPY . /app/

# Expose the port 8000
EXPOSE 8000

# Run the command to start the Django development server
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

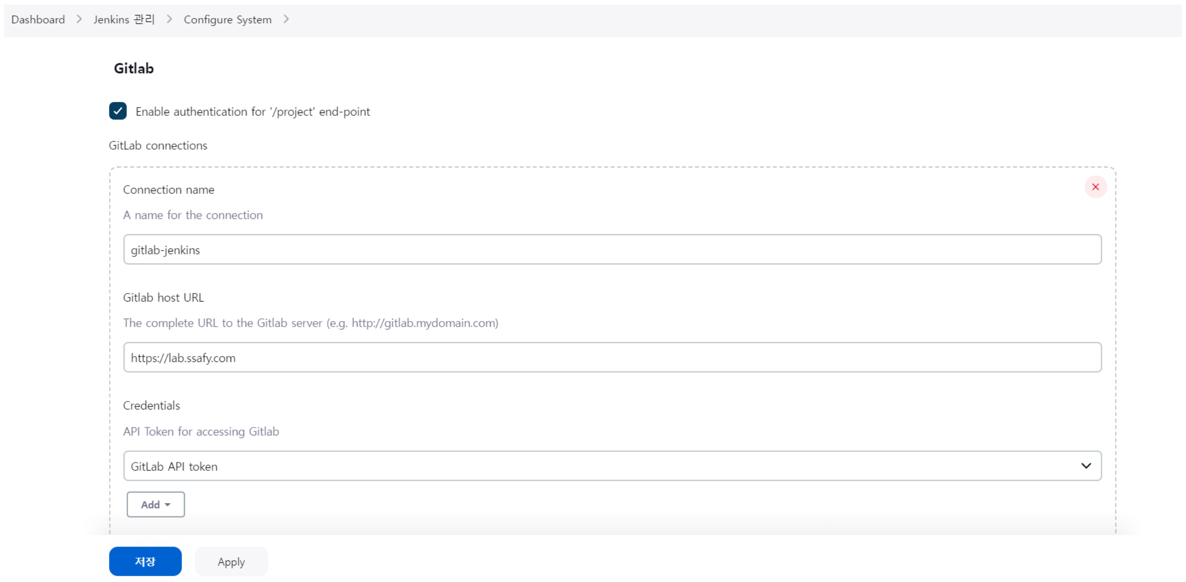
▼ fastAPI container

▼ Dockerfile

```
FROM python:3.7

COPY ./app /app
WORKDIR /app

RUN pip install -r requirements.txt

EXPOSE 80

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]
```

▼ gitlab webhook - jenkins pipeline 연동

▼ gitlab jenkins 연결



Gitlab

✓ Enable authentication for '/project' end-point

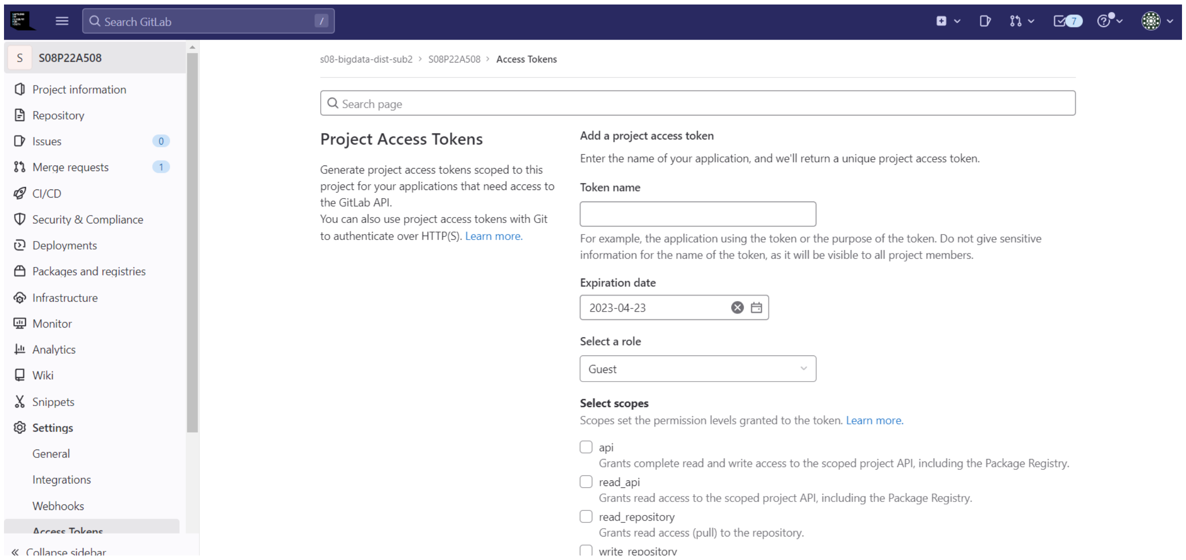
GitLab connections

Connection name
A name for the connection
gitlab-jenkins

Gitlab host URL
The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)
https://lab.ssafy.com

Credentials
API Token for accessing Gitlab
GitLab API token

credentials 은 git lab 의 access token을 발급받아서 등록하기



Project Access Tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API.
You can also use project access tokens with Git to authenticate over HTTP(S). Learn more.

Add a project access token

Enter the name of your application, and we'll return a unique project access token.

Token name

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

Expiration date: 2023-04-23

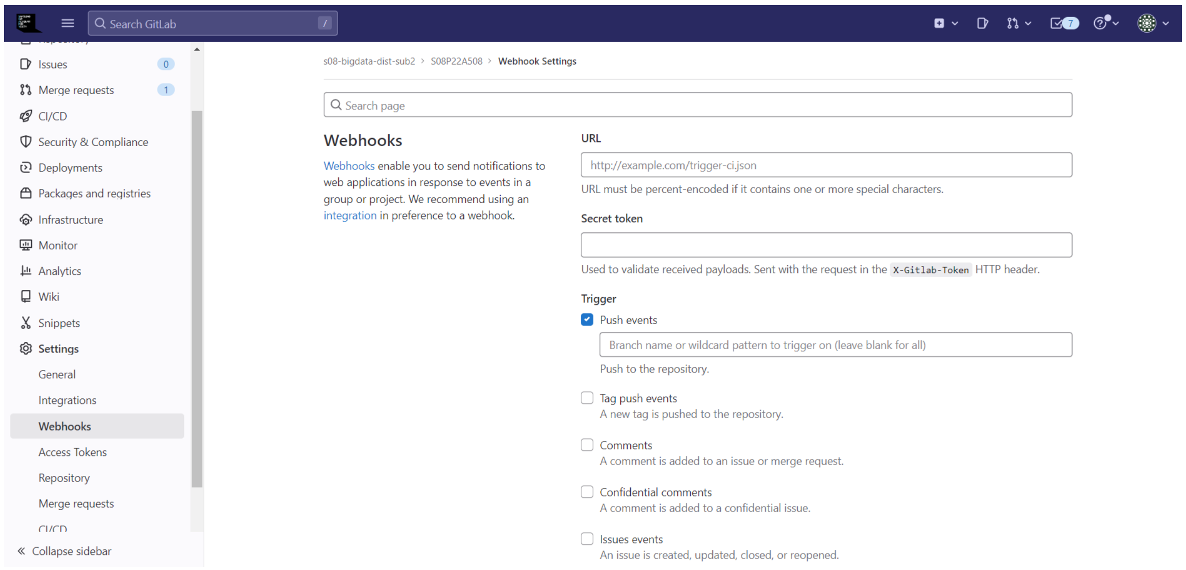
Select a role: Guest

Select scopes

Scopes set the permission levels granted to the token. Learn more.

- [ ] api
  Grants complete read and write access to the scoped project API, including the Package Registry.
- [ ] read_api
  Grants read access to the scoped project API, including the Package Registry.
- [ ] read_repository
  Grants read access (pull) to the repository.
- [ ] write_repository

▼ gitlab webhook 설정



Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a webhook.

URL: http://example.com/trigger-ci.json

URL must be percent-encoded if it contains one or more special characters.

Secret token

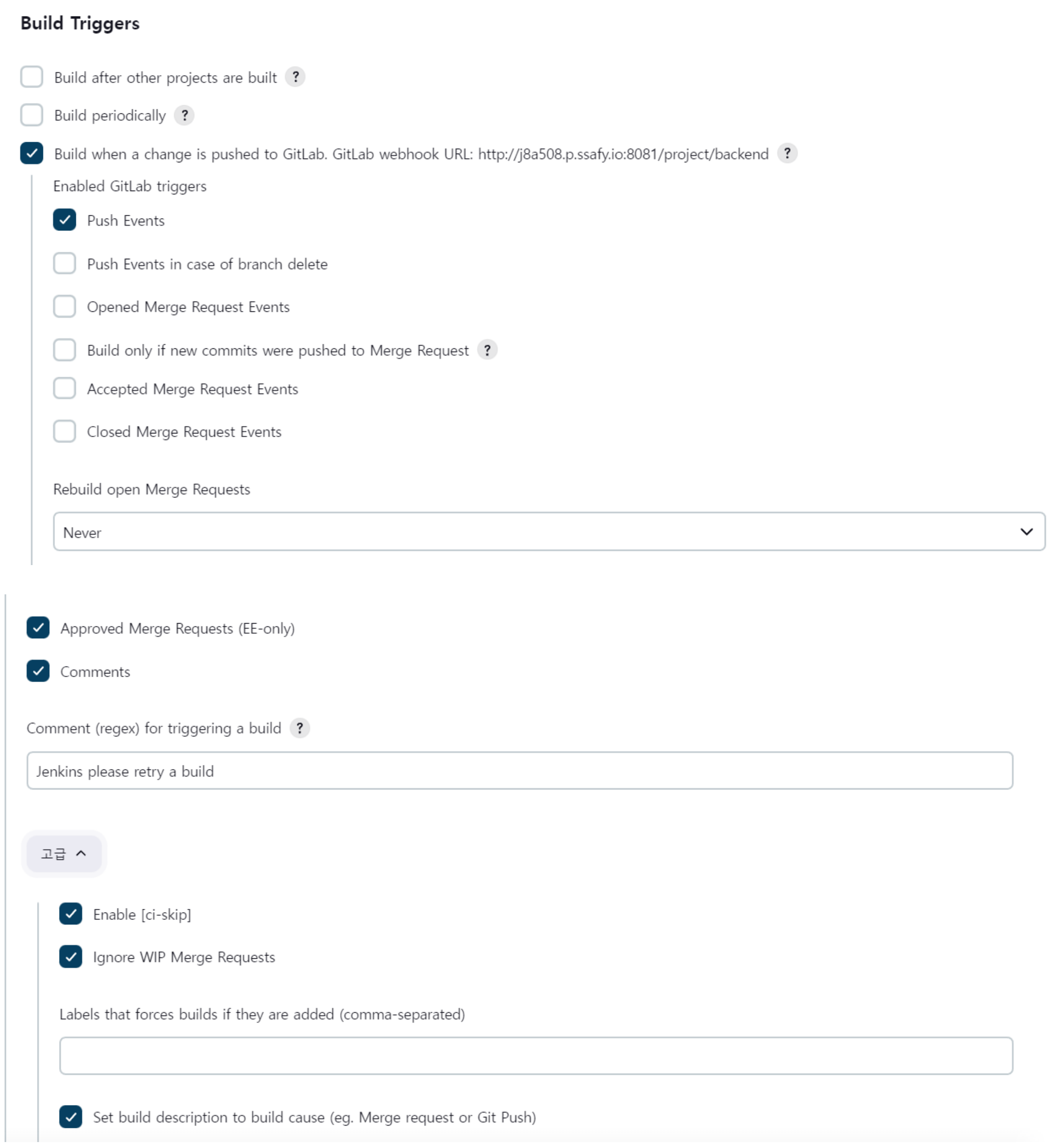
Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

- [x] Push events
  Branch name or wildcard pattern to trigger on (leave blank for all)
  Push to the repository.
- [ ] Tag push events
  A new tag is pushed to the repository.
- [ ] Comments
  A comment is added to an issue or merge request.
- [ ] Confidential comments
  A comment is added to a confidential issue.
- [ ] Issues events
  An issue is created, updated, closed, or reopened.

- url은 젠킨스에서 복붙에서 넣고 , token도 복붙해서 넣기

▼ pipeline 설정하기 : backend



Build Triggers

- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [✓] Build when a change is pushed to GitLab. GitLab webhook URL: http://j8a508.p.ssafy.io:8081/project/backend ?

Enabled GitLab triggers

- [✓] Push Events
- [ ] Push Events in case of branch delete
- [ ] Opened Merge Request Events
- [ ] Build only if new commits were pushed to Merge Request ?
- [ ] Accepted Merge Request Events
- [ ] Closed Merge Request Events

Rebuild open Merge Requests: Never

- [✓] Approved Merge Requests (EE-only)
- [✓] Comments

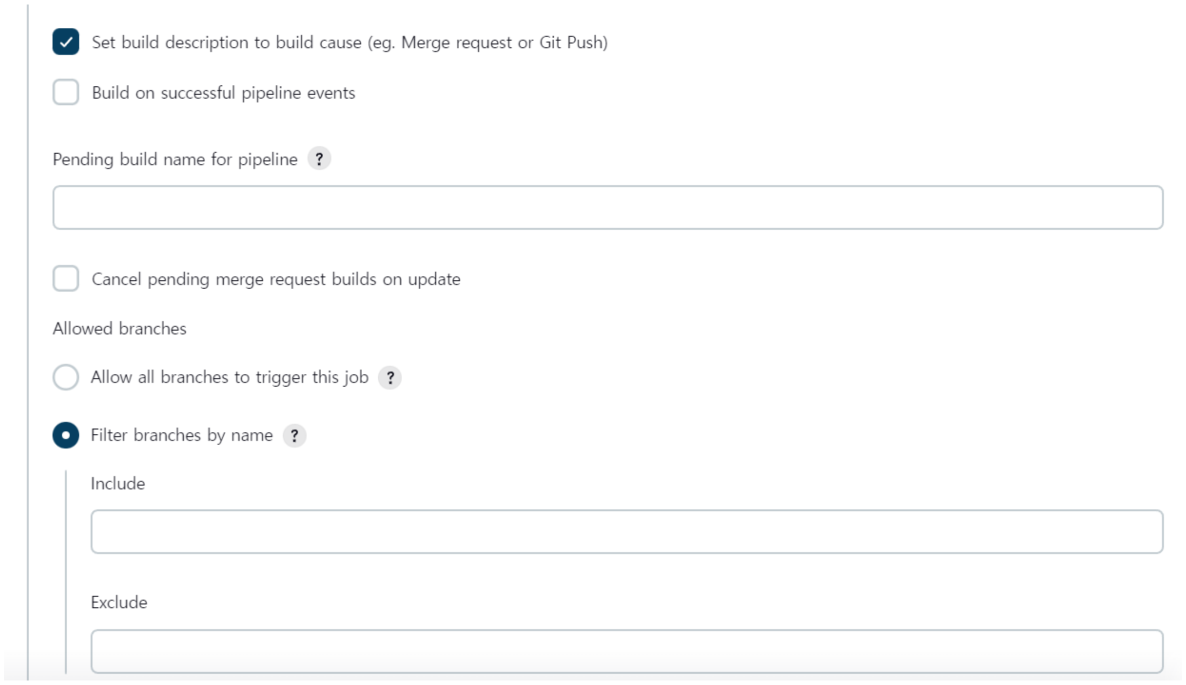
Comment (regex) for triggering a build ?: Jenkins please retry a build

고급

- [✓] Enable [ci-skip]
- [✓] Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated):

- [✓] Set build description to build cause (eg. Merge request or Git Push)


[✓] Set build description to build cause (eg. Merge request or Git Push)

[ ] Build on successful pipeline events

Pending build name for pipeline

[ ] Cancel pending merge request builds on update

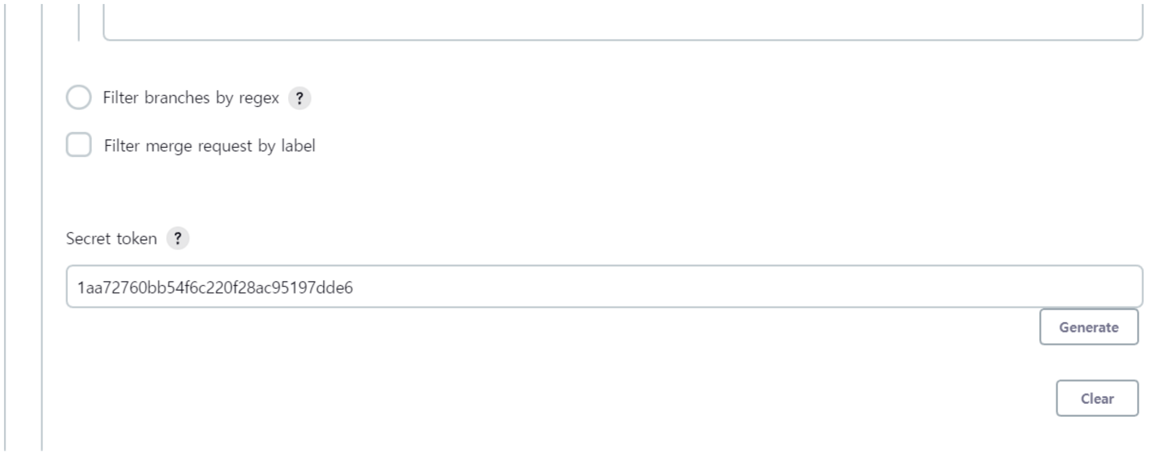
Allowed branches

○ Allow all branches to trigger this job

● Filter branches by name

Include

Exclude


○ Filter branches by regex

[ ] Filter merge request by label

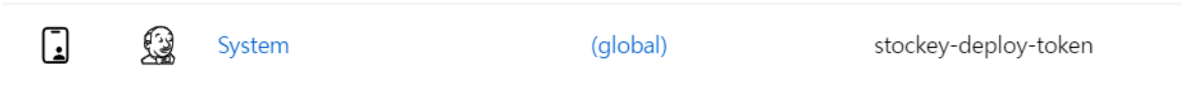
Secret token

1aa72760bb54f6c220f28ac95197dde6

```
pipeline {
    agent any

    environment {
        GIT_URL = "https://lab.ssafy.com/s08-bigdata-dist-sub2/S08P22A508"
    }

    stages {
        stage('Pull') {
            steps {
                git url: "${GIT_URL}", branch: "develop-be", poll: true, changelog: true, credentialsId: 'stockey-deploy-token'

            }
        }

        stage('GradleBuild') {
            steps {
                dir('backend'){
                    sh 'chmod +x gradlew'
                    sh './gradlew clean bootJar'
                }
            }
        }
        stage('Build') {
            steps {
                dir('backend'){
                    sh 'docker stop spring-container || true && docker rm spring-container || true'
                    sh 'docker build --force-rm -t spring-server:1.0.0 .'
                }
            }
        }

        stage('Deploy') {
            steps{
                dir('/home/ubuntu/spring') {
                    sh 'docker run -d -P --name spring-container --env-file ./.env spring-server:1.0.0'
                }
            }
        }

       stage('Finish') {
            steps{
                sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
            }
        }
    }
}
```

- credentialsId는 gitlab에서 deploy-token 발급받아서 credentials에 등록해서 사용

```
<access tokens>
uFozfDFqfk7iyKXSW_LB

<Deploy tokens>
gitlab+deploy-token-5163
KxdAe3vrybiK-F2sezRT
```


System (global) stockey-deploy-token

```
password and username 선택
username
develop-token
id
```

▼ pipeline 설정하기 : frontend

- do not allow concurrent builds 체크
- Build when a change is pushed to GitLab. GitLab webhook URL: http://j8a508.p.ssafy.io:8081/project/frontend? 체크
- gitlab에서 webhook 탭 들어가서 webhook 추가 url이랑 key 넣고 생성
- 파이프라인 작성 후 저장

```
pipeline { // 파이프라인의 시작
    agent any

    environment {
        GIT_URL = "https://lab.ssafy.com/s08-bigdata-dist-sub2/S08P22A508"
    }

    stages {
        stage('Pull') {
            steps {

                git url: "${GIT_URL}", branch: "develop-fe", poll: true, changelog: true, credentialsId: 'stockey-deploy-token'
            }
        }
        stage('Stop and Remove Old Container - Front') {
            // 안되면 try catch 적용해보기
            steps {
                script{
                    try {
                        sh 'docker stop $(docker ps -q --filter ancestor=react-server)'
                        sh 'docker rm $(docker ps -a -q --filter ancestor=react-server)'
                    } catch (Exception e) {
                        echo "An error occurred: ${e}"
                    }
                }
            }

            post {
            success {
                  echo 'Stop and Remove success!'
            }
          }
        }

        stage('Bulid Frontend') {
          // 도커 빌드
         // agent any
          steps {
            echo 'Build Frontend'

            dir ('frontend'){
                sh """
                docker build --force-rm -t react-server .
                """
            }
          }
        post {
            // steps 끝나면 post온다
            // 빌드하다 실패하면 error 뱉고, 나머지 과정 하지말고 종료
            failure {
              error 'This pipeline stops here...'
            }
          }
        }
       stage('Deploy New Frontend Container') {
            steps {
                sh 'docker run --network ssafy-net --name react-container -p 3000:3000 -d react-server'
            }

            post {
            success {
                  echo 'Deploy Frontend success!'
            }
          }
        }

        stage('Finish') {
            steps{
                sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
            }
        }
    }
}
```

▼ pipeline 설정하기 : django

```
pipeline {
    agent any

    environment {
        SECRETS_JSON = credentials('SECRETS_JSON')
        GIT_URL = 'https://lab.ssafy.com/s08-bigdata-dist-sub2/S08P22A508'
    }

    stages {
        stage('Pull') {
            steps {
                git url: "${GIT_URL}", branch: '브랜치 이름', poll: true, changelog: true, credentialsId: '토큰 id'
            }
        }
        stage('Build image') {
            steps {
                dir('djangoServer') {
                    sh 'docker stop django-container || true && docker rm django-container || true'
                    sh 'docker build -t django-server:1.0.0 .'
                }
            }
        }
        stage('Create secrets.json file') {
            steps {
                dir('djangoServer') {
                    sh 'echo $SECRETS_JSON > secrets.json'
                    sh 'cat secrets.json' // Optional: Verify that the file was created correctly
                }
            }
        }
        stage('Deploy') {
            steps {
                sh 'ls'
                sh 'cat $PWD/djangoServer/secrets.json'
                sh 'docker run -d -p 8082:8000  --network ssafy-net --name django-container  -v  /jenkins/workspace/django-server/d
            }
        }
    }
}
```

▼ ssl 설정

▼ 인증서 발급

```
$sudo apt get install letsencrypt
$sudo letsenc rypt certonly standalone d www 제외한 도메인 이름
이메일 작성 후 Agree
뉴스레터 수신 여부 Yes/No
$ssl_certificate /etc/letsencrypt/live/ 도메인이름 /fullchain.pem;
$ssl_certificate_key /etc/letsencrypt/live/ 도메인이름 /privkey.
```

▼ docker-compose

```
version: "3.7"
services:
  certbot:
    image: certbot/certbot:latest
    container_name: cmd_certbot
    command: certonly --webroot --webroot-path=/var/www/certbot --email rlawldud335@naver.com --agree-tos --no-eff-email -d j8a508.
    volumes:
      - /home/ubuntu/nginx/ssl:/etc/letsencrypt:rw
      - /home/ubuntu/nginx/logs:/var/log/letsencrypt:rw
      - /home/ubuntu/nginx/certbot:/var/www/certbot:rw
```

6. 소셜 로그인

1. https://developers.kakao.com/console/app 앱등록
   a. key 발급
      - 네이티브 앱 키: Kakao SDK for Android 초기화, Kakao SDK for iOS 초기화 시 사용
      - REST API 키: REST API 요청 시 HTTP 헤더(Header)에 전달
      - JavaScript 키: Kakao SDK for JavaScript 초기화 시 사용
      - Admin 키: 일부 관리자 기능에 사용, 모든 권한을 갖고 있는 키이므로 유출되지 않도록 주의

2. 플랫폼 등록

a. 도메인 주소

```
http://localhost:3000/oauth/kakao
http://<주소>/oauth/kakao
https://localhost:3000/oauth/kakao
https://<주소>/oauth/kakao
```

b. OpenID Connect 활성화 설정

c. 카카오 로그인 활성화 설정

d. Redirect Url http://localhost:3000/oauth/kakao/callback 설정(본인 서비스에 설정한 URL으로 설정)

- 동의항목 설정가능 (받아올 정보)

3. REST_API_KEY 와 REDIRECT_URI 는 따로 env파일을 만들어서 관리

```
REACT_APP_KAKAO_REST_API_KEY = '<REST_API_KEY>'
REACT_APP_KAKAO_REDIRECT_URI ='<REDIRECT_URI>'
REACT_APP_REST_DEFAULT_URL = '<DEFAULT_URL>'
```

인가코드는 백엔드와 통신해서 전달한다.

2. 데이터 수집 / 적재 / 처리

1. 크롤링

A. 네이버 검색을 통한 뉴스 크롤링 (100대기업, 산업)

run_crawler.py을 통해 news_crawling_complete.py 돌리기

```
import sys
import subprocess as sp
import calendar


if len(sys.argv) < 3:
    print("Usage: python run_crawler.py [start YYYY] [end YYYY]")
    exit(1)
else:
    start_year = int(sys.argv[1])
    end_year = int(sys.argv[2])

    if start_year > end_year:
        print("Year argument must be [start YYYY] <= [end YYYY]")
        exit(1)

    print(f"{start_year} ~ {end_year} year crawling start...")

    for cur_year in range(start_year, end_year + 1):
        for cur_month in range(3, 4):
            cur_month_last_day = calendar.monthrange(cur_year, cur_month)[1]
            # year month start end
            # sp.call(f'python news_crawling_complete.py {cur_year} {cur_month} 30 {cur_month_last_day}')
            sp.call(['python', 'news_crawling_complete.py', str(cur_year), str(cur_month), '1', str(cur_month_last_day)])
```

```
# 1. 라이브러리 불러오기
import os.path
import time

# 크롤링시 필요한 라이브러리 불러오기
from bs4 import BeautifulSoup
import requests
import re
import datetime
from tqdm import tqdm
from datetime import datetime
import pandas as pd
import random
import sys


# 1.1 사용할 변수들

# ConnectionError방지
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/98.0.4718.101"}
ZERO_TOLERANCE = 3  # 끝 페이지를 판단하기 위해서 쓰는 변수. 세번 이상 아무 데이터가 나오지 않으면 다음 날짜로 바꾸기
TITLE_DUP_TOLERANCE = 3  # 타이틀 중복이 세번 이상 나오면 다음 날짜로 진행하기 위한 변수

MOD_NUM = 2  # 페이지당 뉴스기사 두개중 하나씩 뽑기

target_year = -1
target_month = -1
target_start_day = -1
target_end_day = -1

print(sys.argv)

if len(sys.argv) != 5:
    print("Usage: python news_crawling_complete.py {year} {month} 1 {month_last_day}")
    exit(1)
else:
    target_year = int(sys.argv[1])
    target_month = int(sys.argv[2])
    target_start_day = int(sys.argv[3])
    target_end_day = int(sys.argv[4])

    if target_year < 0 or target_month < 0 or target_start_day < 0 or target_end_day < 0:
        print("Usage: python news_crawling_complete.py {year} {month} 1 {month_last_day}")
        exit(1)

# 검색 목록을 csv 파일로 받는다 (하나의 컬럼으로 된 csv파일, 컬럼 명은 상관 없지만 있어야 함)
input_csv_path = 'input/Top100Companies.csv'

# output dir
output_dir = './output'

#################################################################################################

# 1.2 검색할 리스트 만들기
df = pd.read_csv(input_csv_path, encoding='cp949')
keywords = df.iloc[:, 0].tolist()


# 2. 크롤링 시 필요한 함수 만들기

# 입력된 수를 1, 11, 21, 31 ...만들어 주는 함수
def makePgNum(num):
    if num == 1:
        return num
    elif num == 0:
        return num + 1
    else:
        return num + 9 * (num - 1)


def makeUrlTest(search, start_pg, end_pg):
    if start_pg == end_pg:
        start_page = makePgNum(start_pg)
        url = "https://search.naver.com/search.naver?where=news&sm=tab_pge&query=" + search + "&start=" + str(
            start_page)
        # print("생성url: ", url)
        return url
    else:
        urls = []
        for i in range(start_pg, end_pg + 1):
            page = makePgNum(i)
            url = "https://search.naver.com/search.naver?where=news&sm=tab_pge&query=" + search + "&start=" + str(page)
            urls.append(url)
        # print("생성url: ", urls)
        return urls


# 크롤링할 url 생성하는 함수 만들기 -> 키워드와 날짜 넣어주기
def makeUrl(keyword, target_date, ds_de, start_pg, end_pg, sort=0):
    print(f"start_pg = {start_pg}")
    urls = []
    for i in range(start_pg, end_pg + 1):
        start_page = makePgNum(start_pg)
        print(f"start_page = {start_page}")
        url = f"https://search.naver.com/search.naver?where=news&query={keyword}&sm=tab_opt&sort={sort}" \
              f"&photo=0&field=0&pd=3&ds={ds_de}&de={ds_de}&docid=&related=0&mynews=0&office_type=0" \
              f"&office_section_code=0&news_office_checked=&nso=so%3Ar%2Cp%3Afrom{target_date}to{target_date}" \
              f"&is_sug_officeid=0&start={start_page}"
        urls.append(url)
    print("생성urls: ", urls)
    return urls


# html에서 원하는 속성 추출하는 함수 만들기 (기사, 추출하려는 속성값)
def news_attrs_crawler(articles, attrs):
    attrs_content = []
    for i in articles:
        attrs_content.append(i.attrs[attrs])
    return attrs_content


# html생성해서 기사크롤링하는 함수 만들기(url): 링크를 반환
def articles_crawler(i):
    # html 불러오기
    original_html = requests.get(i, headers=headers)
    html = BeautifulSoup(original_html.text, "html.parser")

    url_naver = html.select(
        "div.group_news > ul.list_news > li div.news_area > div.news_info > div.info_group > a.info")  # html
    url = news_attrs_crawler(url_naver, 'href')
    return url


# 3. 크롤링할 네이버 뉴스 URL 추출하기

# 제목, 링크, 내용 1차원 리스트로 꺼내는 함수 생성
def makeList(newlist, content):
    for i in content:
        for j in i:
            newlist.append(j)
    return newlist


def run_crawl_by_date(year, month, start_day, end_day):
    for keyword in keywords:
        for day in tqdm(range(start_day, end_day + 1)):
            zero_cnt = 0  # 정보가 나오지 않는 카운트. ZERO_TOLERANCE를 넘기면 끝 페이지라고 판단하고 다음 날짜로 간다

            title_dup_cnt = 0  # 현재 날짜에 중복 title이 몇개 있는지 체크하는 변수
            title_set = set()  # 특정 날짜의 title
            title_dup_trigger = False

            # 특정 day범위에 대해 page 단위로 가져와
            ################ 날짜 범위 정하기 ################
            date_time_obj = datetime(year=year, month=month, day=day)
            target_date = date_time_obj.strftime("%Y%m%d")
            ds_de = date_time_obj.strftime("%Y.%m.%d")
            ###############################################

            # for loop -> page에 대해서

            for cur_pg in range(1, 401):  # 네이버 뉴스는 400 페이지 까지 지원
                # 뉴스 크롤러 실행
                news_url = []  # 네이버 뉴스 url과 다른 뉴스 url이 혼재되어 있음

                ####### 결과 데이터 담는 리스트 #######
                news_titles = []
                news_contents = []
                news_dates = []
                ###################################
```

```
urls = makeUrl(keyword, target_date, ds_de, cur_pg, cur_pg)  # 1 페이지씩 크롤링 및 저장 (페이지에 대한 url)

for i in urls:
    url = articles_crawler(i)
    news_url.append(url)  # 네이버 뉴스 url과 다른 뉴스 url 구별없이 적재

# 1차원 리스트로 만들기(내용 제외) -> [[pg1/3], [pg2/3], [pg3/3]] 형태를 풀어서 [  ] 1차원 리스트로 풀어주겠다
news_url_1 = []
makeList(news_url_1, news_url)

# NAVER 뉴스만 남기기
naver_urls = []
for i in range(len(news_url_1)):
    if "news.naver.com" in news_url_1[i]:
        naver_urls.append(news_url_1[i])
    else:
        pass

# NAVER sport 기사 제거
final_urls = [x for x in naver_urls if "sports.news" not in x]

# 데이터 볼륨 축소를 위해 일부 뉴스만 남기기
final_url_len = len(final_urls)

news_sample_cnt = int(final_url_len/MOD_NUM)  # 있다면 일부만 크롤링
if news_sample_cnt == 0:  # 크롤링 대상이 너무 적다면 샘플링 하지 않고 그대로 진행
    pass
else:
    final_urls = random.sample(final_urls, news_sample_cnt)  # 그렇지 않다면 샘플링 진행


# 4.뉴스 본문 및 날짜 크롤링하기
# 뉴스 내용 크롤링

for i in tqdm(final_urls):
    # 각 기사 html get하기
    news = requests.get(i, headers=headers)
    news_html = BeautifulSoup(news.text, "html.parser")
    # 뉴스 제목 가져오기
    title = news_html.select_one("#ct > div.media_end_head.go_trans > div.media_end_head_title > h2")
    if title == None:
        title = news_html.select_one("#content > div.end_ct > div > h2")

    # if title in title_set:  # 기존에 크롤링 완료한 title이라면 끝까지 다 간거라서 다음 날짜로 진행하기
    #     trigger = True
    #     break
    # title_set.add(title)

    # 뉴스 본문 가져오기
    content = news_html.select("div#dic_area")
    if content == []:
        content = news_html.select("#articeBody")

    # 기사 텍스트만 가져오기
    # list합치기
    content = ''.join(str(content))

    # html태그제거 및 텍스트 다듬기
    pattern1 = '<[^>]*>'
    title = re.sub(pattern=pattern1, repl='', string=str(title))
    content = re.sub(pattern=pattern1, repl='', string=content)
    pattern2 = """[\n\n\n\n\n// flash 오류를 우회하기 위한 함수 추가\nfunction _flash_removeCallback() {}"""
    content = content.replace(pattern2, '')

    news_titles.append(title)
    news_contents.append(content)

    try:
        html_date = news_html.select_one(
            "div#ct> div.media_end_head.go_trans > div.media_end_head_info.nv_notrans > div.media_end_head_info_datestamp >
        news_date = html_date.attrs['data-date-time']
    except AttributeError:
        news_date = news_html.select_one("#content > div.end_ct > div > div.article_info > span > em")
        news_date = re.sub(pattern=pattern1, repl='', string=str(news_date))
    # 날짜 가져오기
    news_dates.append(news_date)

print(f'day: {day}')
print('news_title: ', len(news_titles))
```

```
                print(f'news_titles = {news_titles}')

                # title_set에 title 데이터 넣기
                for tmp_title in news_titles:
                    if tmp_title in title_set:  # 타이틀 세트에 이 타이틀이 존재한다면
                        title_dup_cnt += 1
                        if title_dup_cnt > TITLE_DUP_TOLERANCE:  # 타이틀 중복이 한도를 넘었다면 다음 날짜로 진행
                            title_dup_trigger = True
                            break
                    else:
                        title_set.add(tmp_title)
                        print(f'title_set = {title_set}')

                # 타이틀 중복이 한도를 넘었다면 다음 날짜로 진행
                if title_dup_trigger:
                    break

                if len(news_titles) == 0 and len(final_urls) == 0:
                    zero_cnt += 1
                    if zero_cnt > ZERO_TOLERANCE:
                        break

                domain_column = [keyword] * len(news_titles)

                # 데이터 프레임 만들기
                news_df = pd.DataFrame(
                    {'date': news_dates, 'domain': domain_column, 'title': news_titles,
                     'content': news_contents, 'url': final_urls})

                # 중복 행 지우기
                news_df = news_df.drop_duplicates(keep='first', ignore_index=True)

                # 데이터 프레임 저장
                now = datetime.now()

                cur_output_dir = os.path.join(output_dir, keyword)

                # output path가 없다면 만들어 주기
                if not os.path.isdir(cur_output_dir):
                    os.makedirs(cur_output_dir)

                save_path = os.path.join(cur_output_dir, f'{now.strftime("%Y%m%d_%H시%M분%S초")}.csv')

                news_df.to_csv(save_path, encoding='utf-8-sig', index=False)
                time.sleep(1.1)


run_crawl_by_date(year=target_year, month=target_month, start_day=target_start_day, end_day=target_end_day)
```

B. 주식 데이터 크롤링

```
import FinanceDataReader as fdr
import pymysql
import math

con = pymysql.connect(host='호스트명', user='유저', password='비밀번호', db='db명', charset='utf8')
cur = con.cursor()

cur.execute("SELECT * FROM stock")

stocks = cur.fetchall()
for stock in stocks:
    print(stock[0], stock[2]) #pk code

    df = fdr.DataReader(stock[2], '2023-01-01', '2023-03-01')

    for row in df.iterrows():
        if  math.isnan(row[1]['Change']) :
            continue
        sql = "INSERT INTO daily_stock(change_rate,close_price,high_price,low_price,open_price,stock_date,volume,stock_id) VALUES(%s, %s, %
        cur.execute(sql, (row[1]['Change'], row[1]['Close'],row[1]['High'], row[1]['Low'], row[1]['Open'], row[0],row[1]['Volume'], stock[0
        con.commit()

con.close()
```

C. 네이버 경제 뉴스 크롤링

```
import subprocess as sp
import sys
import calendar
import os

# 기간내 date 뽑아오기
def make_date(date):
    """
    2022-01-01 ~ 2022-01-05 => 2022-01-01 , 2022-01-02, 2022-01-03, 2022-01-04,2022-01-05
    """
    mate_dates = []
    for year in range(date['start_year'], date['end_year'] + 1):
        if date['start_year'] == date['end_year']:
            target_start_month = date['start_month']
            target_end_month = date['end_month']
        else:
            if year == date['start_year']:
                target_start_month = date['start_month']
                target_end_month = 12
            elif year == date['end_year']:
                target_start_month = 1
                target_end_month = date['end_month']
            else:
                target_start_month = 1
                target_end_month = 12

        for month in range(target_start_month, target_end_month + 1):
            if date['start_month'] == date['end_month']:
                target_start_day = date['start_day']
                target_end_day = date['end_day']
            else:
                if year == date['start_year'] and month == date['start_month']:
                    target_start_day = date['start_day']
                    target_end_day = calendar.monthrange(year, month)[1]
                elif year == date['end_year'] and month == date['end_month']:
                    target_start_day = 1
                    target_end_day = date['end_day']
                else:
                    target_start_day = 1
                    target_end_day = calendar.monthrange(year, month)[1]

            for day in range(target_start_day, target_end_day + 1):
                if len(str(month)) == 1:
                    month = "0" + str(month)
                if len(str(day)) == 1:
                    day = "0" + str(day)

                # 날짜별로 Page Url 생성
                crawling_date = str(year) +"-"+ str(month) +"-"+ str(day)
                mate_dates.append(crawling_date)
    return mate_dates


start_date = sys.argv[1]
end_date = sys.argv[2]

start = list(map(int, start_date.split("-")))
end = list(map(int, end_date.split("-")))

# Setting Start Date
start_year, start_month, start_day = start

# Setting End Date
end_year, end_month, end_day = end

args = [start_year, start_month, start_day, end_year, end_month, end_day]

date = {'start_year': 0, 'start_month': 0, 'start_day' : 0, 'end_year': 0, 'end_month': 0, 'end_day':0}
for key, value in zip(date, args):
    date[key] = value

dates = make_date(date)

# 각 날짜별로 for문 돌아 크롤링 실행
for date in  dates:
    sp.run(["python","korea_economy.py",date,date])
```

D. 기업 설명 크롤링

```
from selenium import webdriver
import datetime
import time
from selenium.webdriver.common.by import By
import pandas as pd
import math
#STEP 1
import pymysql

browser = webdriver.Chrome()

original_window = browser.current_window_handle


def company_crawling(url):
    ## 회사 설명 상단 크롤링
    browser.get(url)
    corp_detail = browser.find_element(By.CLASS_NAME,'corp_detail_box')
    tags = corp_detail.find_element(By.CLASS_NAME,'tag')
    result = {}

    print('='*20,"상단 크롤링",'='*20)
    line = tags.text
    split_tag = line.split('#')
    economy_tag = ''
    for tag in split_tag:
        if tag.startswith("재무평가"):
            economy_rate = tag.split("_")[1]
            economy_tag = economy_rate

    result['basic_info'] = economy_tag



    # 테이블 크롤링
    print('='*20,"테이블 크롤링",'='*20)
    dict = {}
    table = corp_detail.find_element(By.CLASS_NAME,'tbl_com1')
    tbody = table.find_element(By.TAG_NAME,"tbody")
    rows = tbody.find_elements(By.TAG_NAME,"tr")
    for row in rows:
        keys=row.find_elements(By.TAG_NAME,"th")
        values=row.find_elements(By.TAG_NAME,"td")
        for i in range(len(keys)):
            dict[keys[i].text] = values[i].text


    result['company_size'] = dict['기업규모']
    try:
        sales = dict['매출액'].split('\n')[0].split(':')[1].replace(" ","")
    except:
        sales = dict['매출액']
    result['sales'] = sales
    result['credit_rank'] = dict['신용등급']


    ## 대표 브랜드 및 사업 구성
    print('='*20,"대표 브랜드 및 사업 구성",'='*20)
    try:
        corp_brand = browser.find_element(By.CLASS_NAME,'list_brand2')
        businesses = corp_brand.find_elements(By.TAG_NAME,"li")
        business_list  = []
        for business in businesses:
            business_dict = {}
            name = business.find_element(By.CLASS_NAME,'t1').text
            description = business.find_element(By.CLASS_NAME,'t2').text
            description = description.replace("'","")
            business_dict['name'] = name
            business_dict['description'] = description
            business_list.append(business_dict)

        result['businesses'] = business_list
    except:
        result['businesses'] =[]
    return result


def get_stock_id(name):
    cur = con.cursor()
    cur.execute("SELECT * FROM stock where name = %s",name)
    result = cur.fetchone()
    return result


def get_stock_names():
    df = pd.read_csv("기업요약.csv")[['종목명','캐치 url',]]
    return df

def create_stock_db(df,name):
    cur = con.cursor()
    sql = f"UPDATE stock SET company_size = '{df['company_size']}' \
        , company_sales = '{df['sales']}' , credit_rank = '{df['credit_rank']}' \
        , basic_info = '{df['basic_info']}'  where name = '{name}'"
    cur.execute(sql)
    con.commit()

    cur.execute(f"SELECT(stock_id) from stock where name='{name}'")
    stock_id = int(cur.fetchone()[0])
    return stock_id


def create_business_db(df,id):
    cur = con.cursor()
    businnesses = df['businesses']
    for business in businnesses:
        print("사업",business, id)
        cur.execute(f"INSERT INTO business(name,description,stock_id) \
                values ('{business['name']}', '{business['description']}','{id}') ")
        con.commit()
    con.commit()

def check_business_exists(id):
    cur = con.cursor()
    cur.execute(f"SELECT * from business where stock_id = '{id}'")
    answer = cur.fetchall()
    if(answer):
        return True
    else:
        return False

def login():
    url = "https://www.catch.co.kr/Member/Login?ReturnURL=%2F"
    browser.get(url)
    time.sleep(1)
    browser.find_element(By.CLASS_NAME,"ico1").click()
    time.sleep(5)
    for window_handle in browser.window_handles:
        if window_handle != original_window:
            browser.switch_to.window(window_handle)
            break
    id = browser.find_element(By.ID,"id")
    pw =browser.find_element(By.ID,"pw")

    # TOOD
    id.send_keys("네이버 id")
    time.sleep(1)
    # TODO
    pw.send_keys("네이버 비밀번호")
    time.sleep(1)
    browser.find_element(By.ID,"log.login").click()
    time.sleep(15)

def start():
    login()
    browser.switch_to.window(original_window)
    df = get_stock_names()
    for i in range(len(df['종목명'])):
        name = df['종목명'].loc[i]
        url = df['캐치 url'].loc[i]
        stock = get_stock_id(name)
        print(i,"크롤링 중... : ",name,url)
        try:
            math.isnan(url)
            url = ""
        except:
            pass
        if(stock and url):
            time.sleep(5)
            data = company_crawling(url)
            id = create_stock_db(data,name)
            isAlready = check_business_exists(id)
            if(not isAlready):
                create_business_db(data,id)


# con = pymysql.connect(host='호스트명', user='이름', password='패스워드',
#                      db='db명', charset='utf8') # 한글처리 (charset = 'utf8')

start()
```

2. HDFS 적재

A. HDFS에 데이터 put

```
import os
import pandas as pd
from pyspark.sql import SparkSession
from dateutil.parser import parse
from datetime import datetime



pd.options.display.max_rows = 10
pd.options.display.max_columns = 10

pd.options.display.max_rows = None
pd.options.display.max_columns = None


# create a SparkSession
# spark = SparkSession.builder.appName("Write to HDFS").getOrCreate()
# create a SparkSession
spark = SparkSession.builder.appName("Write to HDFS").config("spark.driver.memory", "2g").getOrCreate()

year = 2022

# set the directory path
# input_dir_path = r'/home/ubuntu/jun/preprocessingCode/inputs/company_2022_input/top100'
input_dir_path = rf'/home/ubuntu/jun/preprocessingCode/inputs/economy_2022_input'



# economy, industry, company
news_type = 'economy'

# output hdfs dir
# /user/j8a508/stockey/news/economy
# /user/j8a508/stockey/news/industry
# /user/j8a508/stockey/news/company
output_hdfs_path = rf'hdfs://ip-172-26-0-222.ap-northeast-2.compute.internal:9000/user/j8a508/stockey_v2/news/{news_type}'

csv_output_dir = rf'/home/ubuntu/jun/preprocessingCode/output/{news_type}_{year}_csv_output'


# create the path (including intermediate directories) if it doesn't exist
os.makedirs(csv_output_dir, exist_ok=True)


def change_date_format(time_data):

    date_frac = time_data.split(' ')
    date_frac[0] = date_frac[0].replace('.', '-').rstrip('-')

    if date_frac[1] == '오전':
        ms_lst = list(map(int, date_frac[2].split(':')))
        ms_lst.append('00')
        ms_lst = list(map(str, ms_lst))
        date_frac[2] = ":".join(ms_lst)

    elif date_frac[1] == '오후':
        ms_lst = list(map(int, date_frac[2].split(':')))
        hh = ms_lst[0] + 12
        if hh > 23:
            hh = 12
        ms_lst[0] = hh
        ms_lst.append('00')
        ms_lst = list(map(str, ms_lst))
        date_frac[2] = ":".join(ms_lst)

    del date_frac[1]
    date_time_str = " ".join(date_frac)

    # convert string to datetime object
    date_obj = datetime.strptime(date_time_str, '%Y-%m-%d %H:%M:%S')

    # convert datetime object back to string with desired format
    new_date_str = datetime.strftime(date_obj, '%Y-%m-%d %H:%M:%S')
    return new_date_str


# get the directory names
dir_names = os.listdir(input_dir_path)
# print(dir_names)

process_cnt = 1
loop_len = len(dir_names)
# iterate over the directory names
for dir_name in dir_names:
    print(f'process: {process_cnt} / {loop_len}')
    # construct the full path of the directory

    full_path = os.path.join(input_dir_path, dir_name)

    # check if the full path is a directory
    if os.path.isdir(full_path):
        print(f"Processing directory: {full_path}")

        # get the CSV files in the directory
        csv_files = [f for f in os.listdir(full_path) if f.endswith('.csv')]
        # print(csv_files)
        # print(f'len(csv_files) = {len(csv_files)}')
        # iterate over the CSV files

        whole_df = pd.DataFrame()
        for csv_file in csv_files:
            # construct the full path of the CSV file
            csv_path = os.path.join(full_path, csv_file)

            # read the CSV file into a pandas DataFrame
            df = pd.read_csv(csv_path)
            whole_df = pd.concat([whole_df, df])

        # 데이터 프레임 전처리

        ## 날짜로 정렬
        whole_df_sorted = whole_df.sort_values(by='date').reset_index(drop=True)
        # print(whole_df_sorted.head())

        ## title가 None인 데이터 확인 및 없애기
        none_mask = whole_df_sorted['title'].str.contains('None', na=False)
        missing_title_rows = whole_df_sorted[none_mask]
        whole_df_sorted_rmNa = whole_df_sorted.drop(missing_title_rows.index)

        # # convert the 'date' column to datetime format -> 탐색 / 처리시 활용
        # whole_df_sorted_rmNa['date'] = pd.to_datetime(whole_df_sorted_rmNa['date'])
        # print(whole_df_sorted_rmNa.dtypes)
        # df_month_grp_lst = [group[1] for group in whole_df_sorted_rmNa.groupby(pd.Grouper(key='date', freq='M'))]
        # print(len(df_month_grp_lst))
        # for month_df in df_month_grp_lst:
        #     print('###################')
        #     print(month_df.shape)
        #     print(month_df.head())

        # print(whole_df_sorted_rmNa.head())


        # Convert timestamp column to datetime format and check for invalid values
        invalid_timestamps = whole_df_sorted_rmNa['date'].apply(lambda x: pd.to_datetime(x, format='%Y-%m-%d %H:%M:%S', errors='coerce')).i
        # print(whole_df_sorted_rmNa[invalid_timestamps].head())

        whole_df_sorted_rmNa.loc[invalid_timestamps, 'date'] = whole_df_sorted_rmNa.loc[invalid_timestamps, 'date'].apply(change_date_forma
        # Select only the rows where the timestamp is invalid
        # whole_df_sorted_rmNa = whole_df_sorted_rmNa[invalid_timestamps]

        # dir_name에 공백이 있다면 제거 (ex. 부품 산업)
        domain_name = dir_name.replace(' ', '')
        # 년도 가져오기
        date_info = whole_df_sorted_rmNa.loc[0, 'date']
        year = date_info.split("-")[0]

        # whole_df_sorted_rmNa["hdfs_id"] = whole_df_sorted_rmNa.apply(lambda row: str(row.name) + row["fruit"], axis=1)

        # print(whole_df_sorted_rmNa.head())
        # exit(0)

        # add a new column 'F' that combines the index and values from columns 'A' and 'B'
        whole_df_sorted_rmNa['hdfs_id'] = whole_df_sorted_rmNa.index.astype(str) + '_' + whole_df_sorted_rmNa['date'].astype(str) + '_' + w

        print(whole_df_sorted_rmNa.head())
        save_dir_name = domain_name + '_' + year
        print(f'save_dir_name = {save_dir_name}...')

        # csv_output_dir
        csv_save_path = os.path.join(csv_output_dir, f'{save_dir_name}.csv')
        whole_df_sorted_rmNa.to_csv(csv_save_path, index=False, encoding='utf-8-sig')

        # continue

        # create a spark dataFrame
        sdf = spark.createDataFrame(whole_df_sorted_rmNa)

        # exit(0)
        save_path = os.path.join(output_hdfs_path, save_dir_name)
        # write the DataFrame to HDFS
        # sdf.write.mode("append").option("permission", "u=rw,g=rw,o=rw").parquet(save_path)
        sdf.write.mode("overwrite").option("permission", "u=rw,g=rw,o=rw").parquet(save_path)

    process_cnt += 1

exit(0)
stop the SparkSession
spark.stop()
```

B. HDFS에서 데이터 get

```
from pyspark.sql import SparkSession
import os

# create a SparkSession
spark = SparkSession.builder.appName("Read from HDFS").getOrCreate()

# specify the HDFS input path (디렉토리 경로)
hdfs_path = 'hdfs://ip-172-26-0-222.ap-northeast-2.compute.internal:9000/user/j8a508/stockey/news/company/포스코케미칼_2022'

# read the data from HDFS into a DataFrame
sdf = spark.read.parquet(hdfs_path)

# display the DataFrame
sdf.show()

# convert the PySpark DataFrame to a Pandas DataFrame
pdf = sdf.toPandas()

# display the Pandas DataFrame
# print(pdf)

# Use str.startswith() to create a boolean mask
mask = pdf['date'].str.startswith('2022-01')

# Apply the mask to the dataframe to filter the data
filtered_df = pdf[mask]

print(filtered_df)

# stop the SparkSession
spark.stop()
```

3. mysql 적재

A. 같은 카테고리의 뉴스 합치기

```
import os
import pandas as pd
from pyspark.sql import SparkSession
from dateutil.parser import parse
from datetime import datetime



pd.options.display.max_rows = 10
pd.options.display.max_columns = 10

pd.options.display.max_rows = None
pd.options.display.max_columns = None


# create a SparkSession
# spark = SparkSession.builder.appName("Write to HDFS").getOrCreate()
# create a SparkSession
# spark = SparkSession.builder.appName("Write to HDFS").config("spark.driver.memory", "2g").getOrCreate()


# set the directory path
# input_dir_path = r'/home/ubuntu/jun/preprocessingCode/inputs/company_2022_input/top100'
input_dir_path = rf'/home/ubuntu/jun/preprocessingCode/inputs/economy_2022_input'


# economy, industry, company
news_type = 'economy'

date_info = '2022'

# output hdfs dir
# /user/j8a508/stockey/news/economy
# /user/j8a508/stockey/news/industry
# /user/j8a508/stockey/news/company
# output_hdfs_path = rf'hdfs://ip-172-26-0-222.ap-northeast-2.compute.internal:9000/user/j8a508/stockey_v2/news/{news_type}'

csv_output_dir = rf'/home/ubuntu/jun/preprocessingCode/output/{news_type}/{date_info}_collect_output'


# create the path (including intermediate directories) if it doesn't exist
os.makedirs(csv_output_dir, exist_ok=True)


def change_date_format(time_data):
    date_frac = time_data.split(' ')
    date_frac[0] = date_frac[0].replace('.', '-').rstrip('-')

    if date_frac[1] == '오전':
        ms_lst = list(map(int, date_frac[2].split(':')))
        ms_lst.append('00')
        ms_lst = list(map(str, ms_lst))
        date_frac[2] = ":".join(ms_lst)

    elif date_frac[1] == '오후':
        ms_lst = list(map(int, date_frac[2].split(':')))
        hh = ms_lst[0] + 12
        if hh > 23:
            hh = 12
        ms_lst[0] = hh
        ms_lst.append('00')
        ms_lst = list(map(str, ms_lst))
        date_frac[2] = ":".join(ms_lst)

    del date_frac[1]
    date_time_str = " ".join(date_frac)

    # convert string to datetime object
    date_obj = datetime.strptime(date_time_str, '%Y-%m-%d %H:%M:%S')

    # convert datetime object back to string with desired format
    new_date_str = datetime.strftime(date_obj, '%Y-%m-%d %H:%M:%S')
    return new_date_str
```

```
# get the directory names
dir_names = os.listdir(input_dir_path)
# print(dir_names)

process_cnt = 1
loop_len = len(dir_names)
# iterate over the directory names
for dir_name in dir_names:
    print(f'process: {process_cnt} / {loop_len}')
    # construct the full path of the directory

    full_path = os.path.join(input_dir_path, dir_name)

    # check if the full path is a directory
    if os.path.isdir(full_path):
        print(f"Processing directory: {full_path}")

        # get the CSV files in the directory
        csv_files = [f for f in os.listdir(full_path) if f.endswith('.csv')]
        # print(csv_files)
        # print(f'len(csv_files) = {len(csv_files)}')
        # iterate over the CSV files

        whole_df = pd.DataFrame()
        for csv_file in csv_files:
            # construct the full path of the CSV file
            csv_path = os.path.join(full_path, csv_file)

            # read the CSV file into a pandas DataFrame
            # df = pd.read_csv("./Article_economy_20220103.csv",names=['date','category','press','title','subtitle','content','url'])
            df = pd.read_csv(csv_path, names=['date','domain','press','title','subtitle','content','url'])
            # df = df.drop('subtitle', axis=1)
            df = df.drop(columns=['subtitle', 'press'])

            whole_df = pd.concat([whole_df, df])


        # 데이터 프레임 전처리
        ## news url이 중복된 데이터 지우기
        print(rf'#1. 중복 제거 전 shape = {whole_df.shape}')
        duplicates = whole_df['url'].duplicated()
        whole_df = whole_df[~duplicates]

        whole_df = whole_df.reset_index(drop=True)
        print(rf'#2. url 중복 제거 후 shape = {whole_df.shape}')

        # print("#1")
        # whole_df['date'] = pd.to_datetime(whole_df['date'])

        ## news title이 중복된 데이터 지우기
        duplicates = whole_df['title'].duplicated()
        whole_df = whole_df[~duplicates]
        whole_df = whole_df.reset_index(drop=True)

        print(rf'#3. title 중복 제거 후 shape = {whole_df.shape}')

        duplicates = whole_df['title'].duplicated()
        if duplicates.all():
            print("news url 중복 존재")


        print(whole_df[whole_df['title'] == '우리은행 예·적금 금리 최고 0.30%p 인상'].shape)

        exit(0)

        # # filter the rows based on the column value
        # filtered_df = whole_df[whole_df['title'] == '지난해 딸기·포도 수출액 첫 1억달러 돌파…역대 최대 기록']


        # # print the filtered dataframe
        # print(filtered_df)
        # print(filtered_df.shape)

        # print("#2")
        # whole_df['date'] = pd.to_datetime(whole_df['date'])

        # # filter the rows based on the year
        # filtered_df = whole_df[whole_df['date'].dt.year == 2023]

        # # print the filtered dataframe
        ## 날짜로 정렬
        whole_df_sorted = whole_df.sort_values(by='date').reset_index(drop=True)
        # print(whole_df_sorted.head())

        ## title가 None인 데이터 확인 및 없애기
        none_mask = whole_df_sorted['title'].str.contains('None', na=False)
        missing_title_rows = whole_df_sorted[none_mask]
        whole_df_sorted_rmNa = whole_df_sorted.drop(missing_title_rows.index)

        # # convert the 'date' column to datetime format -> 탐색 / 처리시 활용
        # whole_df_sorted_rmNa['date'] = pd.to_datetime(whole_df_sorted_rmNa['date'])
        # print(whole_df_sorted_rmNa.dtypes)
        # df_month_grp_lst = [group[1] for group in whole_df_sorted_rmNa.groupby(pd.Grouper(key='date', freq='M'))]
        # print(len(df_month_grp_lst))
        # for month_df in df_month_grp_lst:
        #     print('####################')
        #     print(month_df.shape)
        #     print(month_df.head())

        # print(whole_df_sorted_rmNa.head())


        # Convert timestamp column to datetime format and check for invalid values
        invalid_timestamps = whole_df_sorted_rmNa['date'].apply(lambda x: pd.to_datetime(x, format='%Y-%m-%d %H:%M:%S', errors='coerce')).i
        # print(whole_df_sorted_rmNa[invalid_timestamps].head())

        whole_df_sorted_rmNa.loc[invalid_timestamps, 'date'] = whole_df_sorted_rmNa.loc[invalid_timestamps, 'date'].apply(change_date_forma

        # Select only the rows where the timestamp is invalid
        # whole_df_sorted_rmNa = whole_df_sorted_rmNa[invalid_timestamps]

        # dir_name에 공백이 있다면 제거 (ex. 부품 산업)
        domain_name = dir_name.replace(' ', '')
        # 년도 가져오기
        date_info = whole_df_sorted_rmNa.loc[0, 'date']
        year = date_info.split("-")[0]

        # whole_df_sorted_rmNa["hdfs_id"] = whole_df_sorted_rmNa.apply(lambda row: str(row.name) + row["fruit"], axis=1)

        # print(whole_df_sorted_rmNa.head())
        # exit(0)

        # add a new column 'F' that combines the index and values from columns 'A' and 'B'
        whole_df_sorted_rmNa['hdfs_id'] = whole_df_sorted_rmNa.index.astype(str) + '_' + whole_df_sorted_rmNa['date'].astype(str) + '_' + w

        print(whole_df_sorted_rmNa.head())
        save_dir_name = domain_name + '_' + year
        print(f'save_dir_name = {save_dir_name}...')

        # csv_output_dir
        csv_save_path = os.path.join(csv_output_dir, f'{save_dir_name}.csv')
        whole_df_sorted_rmNa.to_csv(csv_save_path, index=False, encoding='utf-8-sig')

        continue

        # create a spark dataFrame
        sdf = spark.createDataFrame(whole_df_sorted_rmNa)

        # exit(0)
        save_path = os.path.join(output_hdfs_path, save_dir_name)
        # write the DataFrame to HDFS
        # sdf.write.mode("append").option("permission", "u=rw,g=rw,o=rw").parquet(save_path)
        sdf.write.mode("overwrite").option("permission", "u=rw,g=rw,o=rw").parquet(save_path)

    process_cnt += 1

exit(0)
# stop the SparkSession
spark.stop()
```

B. 뉴스 적재

```
import os
import pandas as pd
import pymysql
```

```
from datetime import datetime


pd.options.display.max_rows = 10
pd.options.display.max_columns = 10

pd.options.display.max_rows = None
pd.options.display.max_columns = None


mysql_table_name = 'news'

# pdf = pd.read_csv("/home/ubuntu/jun/preprocessingCode/inputs/real종목산업v3.csv", encoding='CP949')

# input_dir = rf'/home/ubuntu/jun/preprocessingCode/output/company_2022_csv_output'
input_dir = rf'/home/ubuntu/jun/preprocessingCode/output/economy/2022_collect_output'
# input_dir = rf'/home/ubuntu/jun/preprocessingCode/output/company/2022_csv_output'

# create connection
conn = pymysql.connect(host='j8a508.p.ssafy.io', user='develop', password='develop', db='stockey_v2')

# create cursor
cursor = conn.cursor()


# get the directory names
file_names = os.listdir(input_dir)
print(file_names)
# exit(0)

loop_len = len(file_names)



print("RDB 적재 시작..")

for file_name in file_names:
    process_cnt = 1
    # construct the full path of the directory

    input_csv_path = os.path.join(input_dir, file_name)
    pdf = pd.read_csv(input_csv_path, encoding='utf-8')
    # print(pdf.head())
    # exit(0)

    num_rows = len(pdf)

    # loop through the rows of the DataFrame
    for index, row in pdf.iterrows():
        print(f'process: {process_cnt} / {loop_len} || row : {index} / {num_rows}')

        cur_date = row.loc['date']
        cur_domain = row.loc['domain']
        cur_title = row.loc['title']
        # cur_content = row.loc['content']
        cur_url = row.loc['url']
        cur_hdfs_id = row.loc['hdfs_id']

        # date 형태 변경
        datetime_obj = datetime.strptime(cur_date, '%Y-%m-%d %H:%M:%S')
        new_date_string = datetime_obj.strftime('%Y%m%d')

        # insert data into MySQL table without inserting data into auto-increment column
        sql = f"INSERT INTO {mysql_table_name} (hdfs_id, news_url, pressed_at, title, category) VALUES (%s, %s, %s, %s, %s)"
        val = (cur_hdfs_id, cur_url, new_date_string, cur_title, cur_domain)

        cursor.execute(sql, val)
        # commit changes
        conn.commit()

    process_cnt += 1



# close connection
conn.close()
```

C. 뉴스에서 키워드 추출 + DB 적재

```python
# -*- coding: utf-8 -*-
import pandas as pd
from keybert import KeyBERT
from transformers import BertModel
from kiwipiepy import Kiwi
from pprint import pprint
import time
import os
import pymysql

ECONOMY = 'economy'
INDUSTRY = 'industry'
STOCK = 'stock'

fileList = ['economy']

news_type = ECONOMY
# news_type = INDUSTRY
# news_type = STOCK

pd.options.display.max_rows = 10
pd.options.display.max_columns = 10

pd.options.display.max_rows = None
pd.options.display.max_columns = None

# Connect to the database
conn = pymysql.connect(host='j8a508.p.ssafy.io', user='develop', password='develop', db='stockey', charset='utf8')
cursor = conn.cursor()

# 명사 추출 함수
kiwi = Kiwi()


def noun_extractor(text):
    results = []
    result = kiwi.analyze(text)
    for token, pos, _, _ in result[0][0]:
        if len(token) != 1 and pos.startswith('N') or pos.startswith('SL'):
            results.append(token)
    return results


# 모델 설정
kw_model = KeyBERT("paraphrase-multilingual-MiniLM-L12-v2")

# 불용어 설정
# stop_words=['kbs', '뉴스', '기자', '속보', '뉴스1', 'mbc', 'sbs', '뉴스데스크', '일보', '올해', '오늘', '내일', '어제', '내년', '하루', '이틀', '사흘',
stop_words = ['코스피', '기업', '한국', '정부', '사업', '경제', '증시', '서울', '코스닥', '가격', '기술', '증권', '개발', '회장', '아파트', '공장', '판매',
              '안전', '장관', '업계', '최대', '뉴욕', '전년', '고객', '분기', '마켓', '도시', '영업', '코리아', '그룹', '회의', '점검', '상반기', '대표',
              '세대', '주간', '세계', '센터', '선정', 'etf', '속도', '증권사', '주가', '추석', '기관', '직원', '외인', '스텝', '평균', '주년', '목표',
              '거래일', '이유', '캠페인', '시작', '자산', '국제', '마감', '문화', '시간', '천억', '신청', '추가', '출발', '주요', '종목', '절반', '특징',
              'vs', '계획', '발언', '지난해', '비용', '인증', '연말', '예산', '회사', '모집', '인터뷰', '오전', '신문', '우리', '시대', '행진', '예측',
              '기록', '충전', '전국', '나스닥', '혜택', '모델', '정보', '결제', '장비', '지급', '취업', '개인', '가치', '진단', '이전', '솔루션', '소통',
              '오후', '상품', '세상', '사진', '초반', '시설', '예약', '스토어', '기차', '이달', '어디', '누구', '위원회', '다음', '제한', '3사', '관리',
              '첫날', '일정', '계열사', '협력사', '내달', '김주현', '선택', '예상', '대응', '지주', '이후', '등록', '홀딩스', '주말', '헤드라인', '후퇴',
              'spc', '창원', '체계', '보고서', '역사', '인천공항', '패키지', '박스', '정상', '강남', 'bnk', '표준', '진행', '은행장', '경험', '필요',
              '타이어', '마이', '컴퍼니', '초청', '충남', '화성', '하늘', '레드', '김대호', '날개', '주민', '도로', '스튜디오', '컨퍼런스', '서울대', '파크',
              '믹스', 'ftx', '대화', '분석', '블루', '바다', '목소리', '실시', '새벽', '연속', '본격', '시험', '지난달', '리포트', '사전', '후보자', '시티',
              '활용', '이상', '농식품부', '카톡', '월급', '메뉴', '재송', '정원', '순위', '참석', '생명', 'dd', '조직', '국민', '접종', '사용', '재계',
              '타운', '지분', '공원', 'kg', '중심', '인사말', '사설', '체크', '케어', '라이브', '플러스', 'fn', '직장인', '비율', '리더십', '바람',
              '컨설팅', '변경', '본부', '닷컴', '출연', '클래스', '근무', '업무', '전면', '전통', '연내', '최종', '용품', '리더', '의무', '비즈니스', '균형',
              '개시', '자동', '향후', '데이', '대비', '집단', '이제', '대신', '수준', '단지', '계속', '구역', '사실', '저장', '테스트', '김동관', '이번',
              '처음', '예고', '확인', '기간', '이하', '여부', '선언', '흐름', '나흘', '이것', '이어', '나라', '아래', '여기', '제외', '지금', '월요일',
              '기기', '칼럼', '대체', '언제', '이동', 'kbs', '뉴스', '기자', '속보', '뉴스1', 'mbc', 'sbs', '뉴스데스크', '일보', '올해', '오늘',
              '내일', '어제', '내년', '하루', '이틀', '사흘', '모레', '작년', '당시', '개월']


# 키워드 추출 함수
def keyword_extractor(content):
    content_nouns = ' '.join(noun_extractor(content))
    keywords = kw_model.extract_keywords(content_nouns, keyphrase_ngram_range=(1, 1), top_n=3, use_mmr=True,
                                         stop_words=stop_words)
    return keywords


import re
```

```
def text_cleaner(text):
    pattern_punctuation = re.compile(r'[^\w\s]')
    return pattern_punctuation.sub('', text).replace('\n', ' ')


def keywd_select(curKeywd):
    keyword_table_check_sql = f"select keyword_id from keyword_v2 where name = %s"
    cursor.execute(keyword_table_check_sql, (curKeywd,))
    return cursor.fetchone()


def keywd_insert(curKeywd):
    keyword_insert_sql = 'insert into keyword_v2 (description, name) values (%s, %s)'
    keyword_insert_data = (None, curKeywd)
    cursor.execute(keyword_insert_sql, keyword_insert_data)
    conn.commit()


def news_relation_insert(newsId, keywordId, industryId, stockId, newsType):
    news_relation_insert_sql \
        = 'insert into news_relation_v2 (industry_id, keyword_id, news_id, stock_id, news_type) values (%s, %s, %s, %s, %s)'
    news_relation_insert_data = (industryId, keywordId, newsId, stockId, newsType)
    cursor.execute(news_relation_insert_sql, news_relation_insert_data)
    conn.commit()


def find_by_industry_id_by_name(industry_name):
    find_by_industry_id_by_name_sql = ""
    cursor.execute(find_by_industry_id_by_name_sql, (industry_name,))
    return cursor.fetchone()


# ================================================================================================#
# article 읽어오기

# keyword_insert_sql = 'insert into keyword_v2 (description, name) values (%s, %s)'
# news_relation_insert_sql = 'insert into news_relation_v2 (industry_id, keyword_id, news_id, stock_id, news_type) values (%s, %s, %s, %s, 

for file in fileList:
    start = time.time()

    # Query the data from the table
    sql = f"SELECT * FROM news where category = '{file}'"
    article_df = pd.read_sql(sql, conn)

    # Print the dataframe
    # print(article_df.head())
    # print(article_df.shape)

    # # insert 할 데이터를 튜플 형태로 append
    # keyword_table_data = []
    # news_relation_table_data = []

    # 모든 기사마다
    for row in article_df.itertuples():
        # Pandas(Index=0, news_id=1, category='economy', hdfs_id='0_2022-01-01 00:00:00_economy',
        # news_url='https://n.news.naver.com/mnews/article/003/0010920824?sid=101', press=None,
        # pressed_at=datetime.date(2022, 1, 1), title='신년사방문규 수은 행장 글로벌 공급망 안정화에 총력')

        # print(row)

        # print(row.hdfs_id)

        text = row.title
        text = text_cleaner(text)  # 공백 및 특수문자 제거
        noun_text = noun_extractor(text)  # 명사만 추출
        keywords = keyword_extractor(text)  # 키워드 추출

        if len(keywords) < 3:
            continue

        rst = [row.hdfs_id, keywords[0][0], keywords[0][1], keywords[1][0], keywords[1][1], keywords[2][0],
               keywords[2][1]]
        # print(">>>>", rst)

        # 뽑은 3개의 키워드
        for i in range(3):
            # 먼저 select 해본다.
            cur_keywd = keywords[i][0]
            res_row = keywd_select(cur_keywd)
            # 현재 키워드 id 저장 변수
            cur_keywd_id = -1

            # 만약 등록된 키워드가 없다면
            if res_row is None:
                # 키워드 insert 후 다시 select 하여 keywd id값을 가져온다.
                keywd_insert(cur_keywd)
                cur_keywd_id = keywd_select(cur_keywd)[0]
            # 만약 등록된 키워드가 있다면
            else:
                # res_row에서 keywd id값을 가져온다.
                cur_keywd_id = res_row[0]

            # news relation 채우기

            if news_type == ECONOMY:
                # 뉴스 id
                news_id = row.news_id
                news_relation_insert(news_id, cur_keywd_id, None, None, ECONOMY)
            elif news_type == INDUSTRY:
                # 뉴스 id

                # 산업 id

                # 주식 id

                # news_relation_insert(newsId, cur_keywd_id, industryId, stockId, INDUSTRY)
                pass
            elif news_type == STOCK:
                # 뉴스 id

                # 산업 id

                # 주식 id

                # news_relation_insert(newsId, cur_keywd_id, industryId, stockId, STOCK)
                pass

    print(file, 'done')
    print(f"{time.time() - start:.4f} sec")

# Close the database connection
cursor.close()
conn.close()
exit(0)
```

D. 일일 키워드 통계 테이블 채우기

```
INSERT INTO keyword_statistic (keyword_id, statistic_date, category, count)
SELECT k.keyword_id AS keyword_id, pressed_at AS statistic_date, 'FREQ' AS category, COUNT(*) AS count
FROM news_relation nr
JOIN news n ON nr.news_id = n.news_id
JOIN keyword k ON nr.keyword_id = k.keyword_id
GROUP BY n.pressed_at, k.keyword_id;
```