



returnz 포팅 문서

☰ Tags	
☰ Text	
🕒 업무	결과물

개발 환경

형상관리

- GitLab

사용 OS

- Windowe
- Ubuntu 20.04

UI/UX

- Figma

DataBase

- Maria-db

FrontEnd

- react
- react qury
- redux
- Tail wind.css
- node 18.12.1

이슈 관리

- Jira

Communitcaion

- Notion
- Mattermost

Server

- AWS EC2

IDE

- Visaial Studio
- IntelliJ

BackEnd

- java11
- gradle
- springSecurity
- redis

프론트 환경 구현

- 래포지토리 clone

```
git clone https://lab.ssafy.com/s08-bigdata-dist-sub2/S08P22C106.git
```

- 내부로 이동

```
cd S08P22C106/front
```

- 연관라이브러리 설치

```
npm i
```

- 개발 서버 실행

```
npm start
```

백엔드 환경 구현

- maria-db 환경 구성 (도커)

```
docker run --name maria-db -d -p 3306:3306 --restart=always -e MYSQL_ROOT_PASSWORD=returnz_c106! mariadb --lower_case_table_names=
```

EC2 maria-db 외부 접속 허용

- maria-db docker 우분투에 접속

```
docker exec -it maria-db bash
```

- 라이브러리 설치 및 수정

```
apt-get update
apt-get install -y vim
cd /etc/mysql/mariadb.conf.d
vim 50-server.cnf
```

```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 127.0.0.1
```

```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address           = 127.0.0.1
```

- 주석 해제 후 mysql 재시작

```
service mysql restart
```

maria-db 개인 계정 및 database 생성

- maria-db 컨테이너에 명령 실행

```
docker exec -it maria-db mysql -u root -p
// 비밀번호 입력
```

- MySQL로 전환

```
use mysql;
```

- 새로운 database 생성

```
create database IF NOT EXISTS `returnz_develop`;
```

- 계정 생성

```
CREATE USER 'user_id'@'%' identified by 'passaord';
```

- 접속을 허용할 User를 Host 정보와 함께 Insert

```
INSERT INTO user (Host, User) VALUES ('%', 'user_id');
```

- MySQL에 대한 모든 권한을 부여

```
GRANT ALL PRIVILEGES ON *.* TO 'user_id'@'%' IDENTIFIED BY 'passaord';
```

- 적용

```
FLUSH PRIVILEGES;
```

백엔드 편집

- 백엔드 경로 이동

```
cd S08P22C106/back
```

- 해당 경로로 ide 실행

redis 구성

- redis 이미지 불러오기

```
docker pull redis
```

- 불러온 이미지를 통해 redis docker 서버 실행

```
docker run --name myredis -d -p 6379:6379 redis
```

빅데이터 분산 환경 구현

구성 세팅의 편의성을 위해 총 9개의 컨테이너로 빅데이터 분산 환경이 구성 되어 있습니다.

▼ 클러스터 세팅

<https://github.com/ManduTheCat/docker-hadoop-spark.git>

- 클론한 위치에 들어가 다음 명령어를 수행한다.

```
docker-compose up -d
```

▼ spark - maria-db 설정

다음 링크에서 커넥터 다운로드

공식 문서

MariaDB ColumnStore with Spark

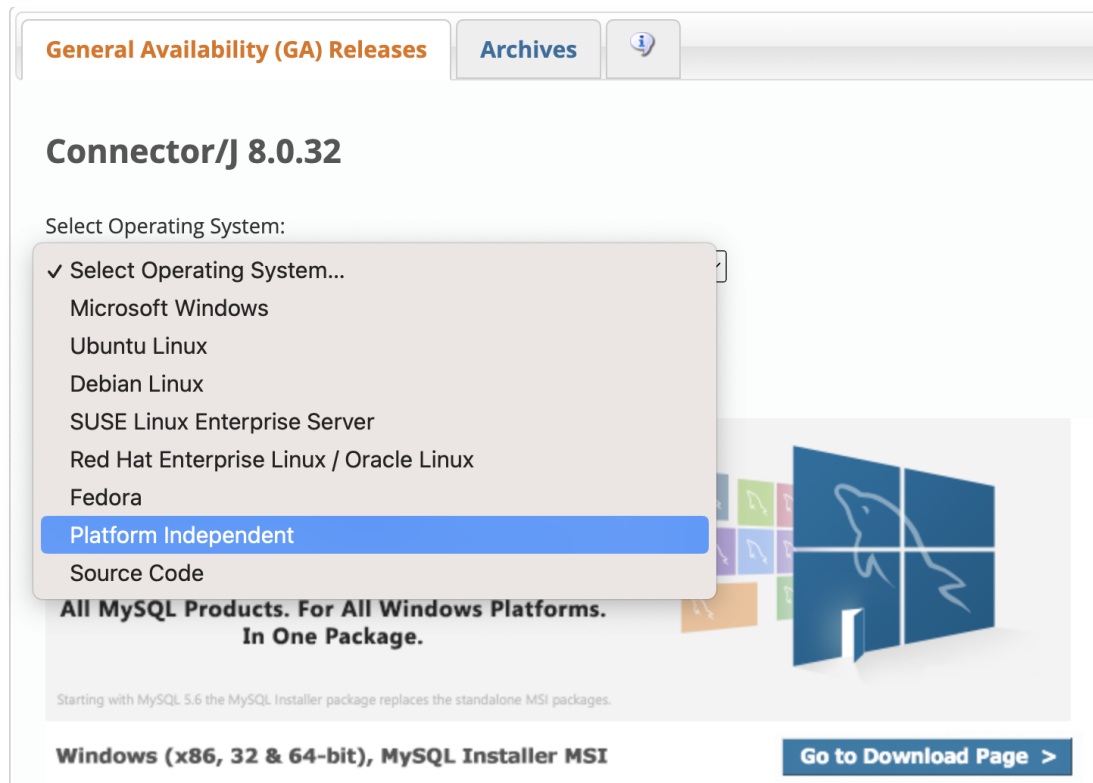
Introduction Apache Spark (<http://spark.apache.org/>) is a popular open source data processing engine. It can be integrated with MariaDB ColumnStore utilizing the Spark SQL fe...

<https://mariadb.com/kb/en/mariadb-columnstore-with-spark/>

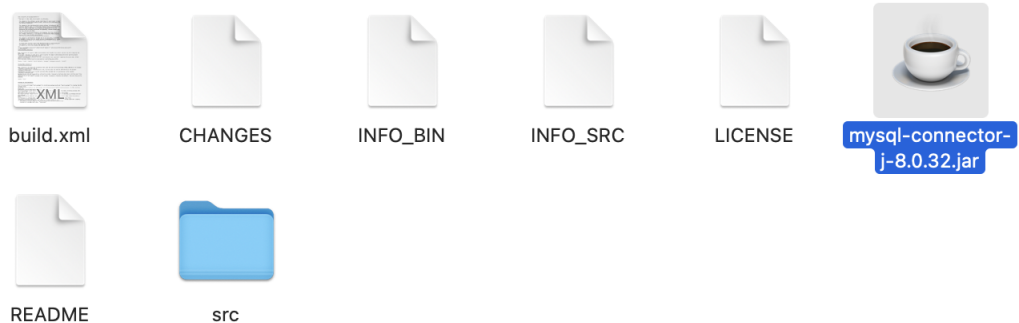
▼ 세팅 방법

1. Mysql Maria Connector 준비

- 다음 링크에서 mysql connector 다운로드
<https://dev.mysql.com/downloads/connector/j/5.1.html>



- 압축 해제



- 다운로드한 커넥터 압출풀고 jar 파일 이동
- 저같은 경우 도커 컨테이너로 spark 구성 하였기에 docker cp 를 사용해 컨테이너 내부로 옮겼습니다
- 자신에 환경에 맞게 spark 내부 jar 폴더에 넣으면됩니다.

```
docker cp mysql-connector-j-8.0.32.jar spark-master:/spark/jar
```

왜 mariaDB 커넥터를 사용하지 않나요?

- 처음에 mariaDB 커넥터를 사용했고 jdbc 를 통해 정상접근 확인.
- 하지만 spark 상에서 `dataframe.show()` 를 사용해 출력하면 데이터 자리에 컬럼명이 들어간체 오는걸로 확인했습니다.

MariaDB 조회 결과

t	nickname	p
	싸피	\$
	싸피2	\$
	싸피3	\$
	NULL	1

Spark application 결과

```
+-----+
|nickname|
+-----+
|nickname|
|nickname|
|nickname|
+-----+
```

- 이 문제를 해결하기 위해 다음 링크를 참조해 지금 문서에 반영했습니다.
<https://stackoverflow.com/questions/40547993/spark-sql-jdbc-returning-only-column-nameshttps://stackoverflow.com/questions/66983401/spark-mariadb-jdbc-sql-query-returns-column-names-instead-of-column-values>

2. Spark Application 작성

Maria db 에 연결 하고 테이블을 읽어오는 pyspark application 입니다.

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MariaDB Connection Example").getOrCreate()

url = 'jdbc:mysql://maria-db:3306/mj_db'

jdbcDF = spark.read.format("jdbc")\
    .option("url", url)\
    .option("user", "yg")\
    .option("password", "yg")\
    .option("dbtable", "member")\
    .load()

jdbcDF.show()
```

다음 코드는 외부 접근이 가능한 디비 유저, 비밀번호 를 입력하면됩니다.

외부 접근되는 계정인 아닌 root 를 사용할경우 경우에 따라 permission 애러나 발생할수 있습니다.

```
.option("user", "user") \\  
.option("password", "pw") \\  

```

작성 후 `saprk/app/` 에 저장했습니다 각자 환경에 맞춰 저장하시면됩니다.

3. Spark-submit 으로 실행

실행에 들어가기 앞서 만약 MariaDB 가 도커 컨테이너로 구성 되어있다면 spark클러스터와 같은 네트워크 에 구성되어 있는지 확인하여야합니다.

연결이 안되 있다면 다음 명령어를 사용해 연결하세요

```
docker network connect 스파크_네트워크 MariaDB_컨테이너이름
```

실행

spark-master 컨테이너 이동 후 다음 환경 변수 추가후

```
export PYTHONIOENCODING=utf8
```

spark/bin 으로 이동 하고 파일 실행

```
./spark-submit --driver-class-path /usr/share/java/mysql-connector-8.0.32.jar ../app/pyspark/readMaria.py
```

```
./spark-submit --driver-class-path ../jars/mysql-connector-8.0.32.jar ../application/read.py
```

```
./spark-submit --driver-class-path ../jars/mysql-connector-8.0.32.jar --jars ../jars/mysql-connector-8.0.32.jar ../application/wr:
```

명령어 설명

- spark-submit 스파크 어플리케이션을 실행하기 위한 명령어
- --driver-class-path /usr/share/java/mysql-connector-8.0.32.jar jdbc 연결한 커넥터 지정
- ../app/pyspark/dbConnect.py 실행할 파일 위치

결과

member_id	account_non_expired	account_non_locked	accumulated_return	enabled	enroll_date	game_count	nickname	password	profile_icon	streak_count	username	state
1	true	true	0	true	2023-03-15 17:29:10	0	새피	\$2a\$10\$hm139aIdw...	A	0	ssafy@naver.com	null
2	true	true	0	true	2023-03-16 17:43:12	0	새피2	\$2a\$10\$S/43YuUZNj...	A	0	ssafy2@naver.com	OFF
3	true	true	0	true	2023-03-17 16:30:47	0	새피3	\$2a\$10\$WNZtD0dArB...	A	0	ssafy3@naver.com	OFF

만약

출력 시 UnicodeEncodeError: 'ascii' codec can't encode character 에러 발생 한다면
해결책 으로 다음 과 같이 환경변수를 추가하면됩니다

```
export PYTHONIOENCODING=utf8
```

<https://stackoverflow.com/questions/39662384/pyspark-unicodeencodeerror-ascii-codec-cant-encode-character>


▼ 제풀린 스파크 설정

1. MySQL 커넥터 다음 경로로 이동

경로는 임의로 만든것이며 원하는 경로에 봐도 됩니다. 다만 다음 단계에서 환경변수 설정시 그경로를 반영해주세요!

다운로드 주소입니다

MySQL :: Download Connector/J

 <https://dev.mysql.com/downloads/connector/j/5.1.html>

- 다운로드 이후에는 편의를 위해 이름을 mysql-connector-8.0.32.jar 바꿔주세요(j를제외)

컨테이너 안에 다음 경로를 만들고

/opt/zeppelin/jdbc-drivers

컨테이너 안으로 전송

```
docker cp mysql-connector-8.0.32.jar spark-zeppelin:/opt/zeppelin/jdbc-drivers
```

2. 설정 및 재시작

재플린 설치 경로로 이동

다음 경로는 우리 환경기준예시

/usr/zeppelin/zeppelin-0.9.0-bin-all

```
cd /usr/zeppelin/zeppelin-0.9.0-bin-all/conf
```

zeppeline-env.sh 파일에 다음 내용추가

```
export ZEPPELIN_JDBC_DRIVER_PATH=/opt/zeppelin/jdbc-drivers
export SPARK_SUBMIT_OPTIONS="--driver-class-path /opt/zeppelin/jdbc-drivers/mysql-connector-8.0.32.jar --jars /opt/zeppelin/jdbc-drivers/mysql-connector-8.0.32.jar"
```

재플린 인터프리터 설정 두가지 추가

환경변수와

spark.jars	
/opt/zeppelin/jdbc-drivers/mysql-connector-8.0.32.jar	

Comma-separated list of Maven coordinates of jars to include on the driver and

아티팩트

Dependencies

Artifact

Exclude

/opt/zeppelin/jdbc-drivers/mysql-connector-8.0.32.jar

다음 명령어로 재플린 재시작

```
./bin/zeppelin-daemon.sh restart
```

다음 코드를 스파크 노트북으로 실행하면 됨

```
%pyspark
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder.appName("MariaDB Connection Example").getOrCreate()

url = 'jdbc:mysql://maria-db:3306/mj_db'

jdbcDF = spark.read.format("jdbc")\
    .option("url", url)\
    .option("user", "yg")\
    .option("password", "yg")\
    .option("dbtable", "member")\
    .load()

jdbcDF.show()
```

```
%pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MariaDB Connection Example").getOrCreate()

url = 'jdbc:mysql://maria-db:3306/mj_db'

jdbcDF = spark.read.format("jdbc")\
    .option("url", url)\
    .option("user", "yg")\
    .option("password", "yg")\
    .option("dbtable", "member")\
    .load()

jdbcDF.show()
```

SPARK JOB FINISHED

member_id	account_non_expired	account_non_locked	accumulated_return	avg_profit	enabled	enroll_date	game_count	nickname	password	profile_icon	statel	streak_count	username
1	true	true	0	0.0	true	2023-03-31 22:38:45	0	ssafy2	\$2a\$10\$FAeMc/9kG...	A	OFF	0	ssafy2@naver.com
2	true	true	0	0.0	true	2023-03-31 22:40:30	0	ssafy3	\$2a\$10\$aydQag.izB...	A	ON	0	ssafy3@naver.com
3	true	true	0	0.0	true	2023-03-31 22:49:56	0	ssafy1	\$2a\$10\$9aewsDsYeP...	A	OFF	0	ssafy1@naver.com
4	true	true	0	0.0	true	2023-04-01 00:07:11	0	moon5	\$2a\$10\$Q7K18kdLVX...	A	OFF	0	moon5@naver.com

CD/CI 환경 구현

- ci 는 gitlab webhcock 과 젠킨스를 활용해 진행했습니다.
- 클린코드 를 위해 소나큐브 를 활용해 코드 정적검사를 수행 했습니다.
- cd 구현은 젠킨스의 ssh publish 를 활용해 백엔드는 nohup , 프론트는 빌드파일을 nginx 로 전달했습니다.

젠킨스 내부설정

1. ssh 설정

- a. 발급받은 pem키, 서버 설정

Publish over SSH

Jenkins SSH Key ?

Passphrase ?

Concealed Change Password

Path to key ?

Key ?

SSH Servers

SSH Server

Name ?

returnz

Hostname ?

j8c106.p.ssafy.io

Username ?

ubuntu

Remote Directory ?

/home/ubuntu/jenkinsBuild

고급 ▾

Test Configuration

추가

2. credentail 설정



a. userNameWithPassword

i. gitlab id password 등록


mandu
krocd@naver.com/*****
Username with password


b. secret key

i. 소나 큐브 프로젝트 키 등록


sonarQube
sonarQube
Secret text


3. sonarQube 서버 설정

4. Global Tool Configuration/Node 18.12.1설정

NodeJS

NodeJS installations Edited

NodeJS installations

List of NodeJS installations on this system

Add NodeJS

NodeJS

Name

returnz

☒ Install automatically ?

Install from nodejs.org

Version

NodeJS 18.12.1

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail
☐ Force 32bit architecture

젠킨스 아이템 설정

젠킨스 파이프라인은 SCM 을 활용 했으며 gitlab repository 안 jenkins 에 존재합니다.

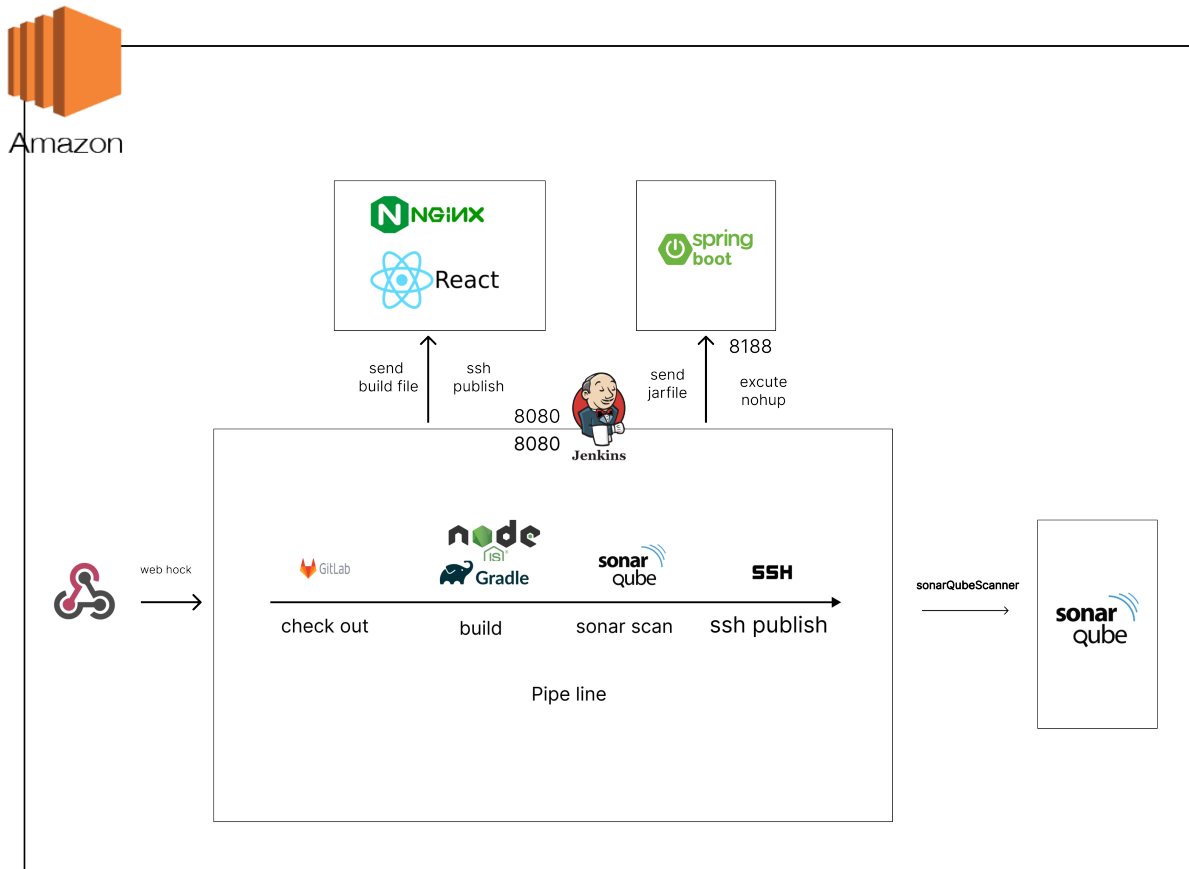
각각 프론트는 `front.jenkinsfile` 백엔드는 `back.jenkinsfile` 로 존재합니다

두개의 젠킨스 아이템이 존재하며 각각 프론트, 백엔드 를 담당합니다.

S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간
		returnz_back	7 hr 33 min #624	20 hr #603	1 min 1 sec
		returnz_front	19 min #587	2 days 11 hr #517	1 min 3 sec

아이콘: S M L Icon legend Atom feed for all Atom feed for failures Atom feed for just latest builds

두 아이템 모두 파이프라인 scm 설정은 동일하게 진행하면 자동으로 git lab 에 있는 파이프라인 스크립트를 읽어오는 형태로 구성 했습니다.



1. 파이프 라인 설정

a. 빌드 트리거 설정

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://j8c106.p.ssafy.io:5001/project/returnz_back ?

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

b. pipeline scm, RepositoryURL, Branches to build(SCM 읽어올 브랜치), Script Path(scm 파일경로) 설정

데이터 수집

Yahoo Query 데이터 수집

▼ Yahoo Query를 통한 데이터 수집 방법

Yahoo Query 데이터 가져오기

- python library 설치

```
pip install yahooquery
pip install yahoofinance
pip install pandas
pip install numpy
```

- valid_company_list.csv를 이용해서 데이터를 가져옴
- valid_company_list.csv는 yahoo query 조회시 얻을 수 있는 데이터를 저장한 파일

```
import pandas as pd
import numpy as np
import yfinance as yf
import datetime
import yahooquery as yq
from yahooquery import Ticker
from datetime import datetime, timedelta

symbol_id = pd.read_csv("valid_company_list.csv", encoding='euc-kr')
symbol_id
```

valid_company_list.csv 파일 형식

Unnamed: 0		symbol	한글 종목명	상장일	시장구분	종목명	업종명	시가총액
0	1	095570.KS	AJ네트웍스보통주	2015-08-21	KOSPI	AJ네트웍스	서비스업	2.345800e+11
1	2	006840.KS	AK홀딩스보통주	1999-08-11	KOSPI	AK홀딩스	기타금융	2.046750e+11
2	3	282330.KS	BGF리테일보통주	2017-12-08	KOSPI	BGF리테일	유통업	3.099000e+12
3	4	027410.KS	BGF보통주	2014-05-19	KOSPI	BGF	기타금융	3.996180e+11
4	5	138930.KS	BNK금융지주보통주	2011-03-30	KOSPI	BNK금융지주	기타금융	2.024060e+12
—	—	—	—	—	—	—	—	—
1150	1151	XEL	Xcel Energy Inc. Common Stock	NaN	NASDAQ	Xcel Energy Inc. Common Stock	Utilities	3.573533e+10
1151	1152	XM	Qualtrics International Inc. Class A Common Stock	2021	NASDAQ	Qualtrics International Inc. Class A Common Stock	Technology	1.049940e+10
1152	1153	ZBRA	Zebra Technologies Corporation Class A Common ...	1991	NASDAQ	Zebra Technologies Corporation Class A Common ...	Technology	1.464624e+10
1153	1154	ZM	Zoom Video Communications Inc. Class A Common ...	2019	NASDAQ	Zoom Video Communications Inc. Class A Common ...	Technology	2.007377e+10
1154	1155	ZS	Zscaler Inc. Common Stock	2018	NASDAQ	Zscaler Inc. Common Stock	Technology	1.584316e+10
1155 rows x 8 columns								

1. asset_profile

- Information related to the company's location, operations, and officers.
- asset_profile이란 dict를 생성한 후 해당 dict에 값을 넣는다.
- 회사에 따라서 존재하지 않는 값들이 있으므로 체크한 후 넣어준다.
- 회사에 대한 정보가 없을 경우 String 형태로 값이 들어온다.

- 특정 값만 없을 경우 해당 key가 빠져있다.

https://yahooquery.dpguthrie.com/guide/ticker/modules/#asset_profile

```
asset_profile = {
    "country" : list(),
    "industry" : list(),
    "sector" : list(),
    "phone" : list(),
    "website" : list(),
}

non_data_symbols = list()

count = 0;
for idx, row in symbol_id.iterrows():
    ticker = yq.Ticker(row["symbol"], backoff_factor=1)

    print(row["symbol"])

    if(type(ticker.asset_profile[row["symbol"]]) is not str):

        if "country" in ticker.asset_profile[row["symbol"]].keys():
            asset_profile["country"].append(ticker.asset_profile[row["symbol"]]["country"])
        else: asset_profile["country"].append("None")

        if "industry" in ticker.asset_profile[row["symbol"]].keys():
            asset_profile["industry"].append(ticker.asset_profile[row["symbol"]]["industry"])
        else: asset_profile["industry"].append("None")

        if "sector" in ticker.asset_profile[row["symbol"]].keys():
            asset_profile["sector"].append(ticker.asset_profile[row["symbol"]]["sector"])
        else: asset_profile["sector"].append("None")

        if "phone" in ticker.asset_profile[row["symbol"]].keys():
            asset_profile["phone"].append(ticker.asset_profile[row["symbol"]]["phone"])
        else: asset_profile["phone"].append("None")

        if "website" in ticker.asset_profile[row["symbol"]].keys():
            asset_profile["website"].append(ticker.asset_profile[row["symbol"]]["website"])
        else: asset_profile["website"].append("None")

    else: non_data_symbols.append(row["symbol"])

    count += 1
```

- 회사 정보가 없는 종목 출력

```
non_data_symbols

# 출력결과
['094800.KS', '446070.KS', '109070.KS', '168490.KS']
```

- DataFrame으로 변환

```
print(len(asset_profile["country"]), len(asset_profile["industry"]), len(asset_profile["sector"]), len(asset_profile["phone"]),
len(asset_profile["website"]))
df_symbol = pd.DataFrame.from_dict(asset_profile)
df_symbol
```

```
# 출력결과
country industry sector phone website
0 South Korea Rental & Leasing Services Industrials 82 2 6363 9999 <https://www.ajurental.com>
1 South Korea Chemicals Basic Materials 82 2 768 2923 <https://www.aekyunggroup.co.kr>
2 South Korea Grocery Stores Consumer Defensive 82 1 577 8007 <https://www.bgfretail.com>
3 South Korea Department Stores Consumer Cyclical 82 1 577 3663 <https://www.bgf.co.kr>
4 South Korea Banks-Regional Financial Services 82 5 1620 3000 <https://www.bnkfg.com>
... ..
1146 United States Utilities-Regulated Electric Utilities 612 330 5500 <https://www.xcelenergy.com>
1147 United States Software-Application Technology 385 203 4999 <https://www.qualtrics.com>
1148 United States Communication Equipment Technology 847 634 6700 <https://www.zebra.com>
1149 United States Software-Application Technology 888 799 9666 <https://www.zoom.us>
1150 United States Software-Infrastructure Technology 408 533 0288 <https://www.zscaler.com>
```

- csv 파일로 저장

2. grading_history

- Data related to upgrades / downgrades by companies for a given symbol(s)
- 한국 데이터는 없음
- 반환 결과가 데이터프레임, 합쳐서 csv 파일로 반환

| https://yahooquery.dpguthrie.com/guide/ticker/modules/#grading_history

```
grading_history= pd.concat([grading_history, ticker.grading_history])
```

3. index_trend

- Trend data related given symbol(s) index, specifically PE and PEG ratios
- pe, peg를 구하기 위해 사용하지만, get_financial_data에서 가져올 수 있으므로 생략

4. get_financial_data

- Obtain specific data from either cash flow, income statement, balance sheet, or valuation measures.
- get_financial_data 사용 시 오류
- valuation_mesuar, balance_sheet, income_statement, cash_flow로 따로 가져올 경우 데이터가 없는 경우가 있음
- all_financial_data를 사용해서 모든 데이터 가져온 후 분류

```
if(type(ticker.all_financial_data()) is pd.DataFrame):  
    financial_data = pd.concat([financial_data, ticker.all_financial_data()])
```

| https://yahooquery.dpguthrie.com/guide/ticker/modules/#index_trend

5. Historical Price

- Retreives historical pricing data (OHLC) for given symbol(s)

6. corporate_events

- Significant events related to a given symbol(s)
- 한국 데이터 없음
- 데이터 프레임 형태로 데이터 제공

```
corporate_events = pd.concat([corporate_events, ticker.corporate_events])
```

7. news

- Get news headline and summary information for given symbol(s)
- 현재 (2023-03-21)기준 Example 수행 시 ['error'] 리턴

```
aapl = Ticker('aapl')  
aapl.news(5)
```

8. recommendations

- Get real-time quote information for given symbol(s)

- 한국 코드도 가능
- 관련 유사 종목으로 사용할 예정

```
recommendations = {
    "symbol_id" : list(),
    "simillar_symbol_id" : list(),
    "score" : list()
}
recommendation_list = list()
for recommend in tickers.recommendations["001360.KS"]["recommendedSymbols"]:
    recommendations["symbol_id"].append("001360.KS")
    recommendations["simillar_symbol_id"].append(recommend["symbol"])
    recommendations["score"].append(recommend["score"])
recommendations
```

Ticker

- 위에서 가져오는 정보들은 대부분 Ticker로 가져옴.
- 중복 호출 방지를 위해서 한번에 처리하는 코드로 변환
- 예러 코드 발생 시 String 형태로 오므로 str인지 비교하는 구문을 생성함.
- count 변수는 의미 없음 (중간 취소 및 확인을 위해 사용)
- Ticker Data 수집

```
asset_profile = {
    "symbol_id" : list(),
    "country" : list(),
    "industry" : list(),
    "sector" : list(),
    "phone" : list(),
    "website" : list(),
}

long_business_summary = {
    "symbol_id" : list(),
    "summary" : list(),
}

recommendations = {
    "symbol_id" : list(),
    "simillar_symbol_id" : list(),
    "score" : list()
}

grading_history = pd.DataFrame()
corporate_events = pd.DataFrame()
financial_data = pd.DataFrame()

non_data_symbols = list()

count = 0;
for idx, row in symbol_id.iterrows():
    ticker = yq.Ticker(row["symbol"], backoff_factor=1)

    print(row["symbol"])

    #### Ticker - Asset Profile
    if(type(ticker.asset_profile[row["symbol"]]) is not str):

        asset_profile["symbol_id"].append(row["symbol"])

        if "country" in ticker.asset_profile[row["symbol"]].keys():
            asset_profile["country"].append(ticker.asset_profile[row["symbol"]]["country"])
        else: asset_profile["country"].append("None")

        if "industry" in ticker.asset_profile[row["symbol"]].keys():
            asset_profile["industry"].append(ticker.asset_profile[row["symbol"]]["industry"])
        else: asset_profile["industry"].append("None")

        if "sector" in ticker.asset_profile[row["symbol"]].keys():
            asset_profile["sector"].append(ticker.asset_profile[row["symbol"]]["sector"])
        else: asset_profile["sector"].append("None")

        if "phone" in ticker.asset_profile[row["symbol"]].keys():
```



```

        asset_profile["phone"].append(ticker.asset_profile[row["symbol"]]["phone"])
    else: asset_profile["phone"].append(None)

    if "website" in ticker.asset_profile[row["symbol"]].keys():
        asset_profile["website"].append(ticker.asset_profile[row["symbol"]]["website"])
    else: asset_profile["website"].append(None)

    ##### Ticker - Long Business Summary
    if "longBusinessSummary" in ticker.asset_profile[row["symbol"]].keys():
        long_business_summary["symbol_id"].append(ticker.asset_profile[row["symbol"]])
        long_business_summary["summary"].append(ticker.asset_profile[row["symbol"]]["longBusinessSummary"])

else: non_data_symbols.append(row["symbol"])

##### Ticker - Grading History
grading_history = pd.concat([grading_history, ticker.grading_history])

### Ticker - Corporate Events
if type(ticker.corporate_events) is not str:
    corporate_events = pd.concat([corporate_events, ticker.corporate_events])

### Ticker - recommendations
if type(ticker.recommendations[row["symbol"]]) is not str:
    for recommend in ticker.recommendations[row["symbol"]]["recommendedSymbols"]:
        recommendations["symbol_id"].append(row["symbol"])
        recommendations["similar_symbol_id"].append(recommend["symbol"])
        recommendations["score"].append(recommend["score"])

# Ticker - Financial Data
if(type(ticker.all_financial_data()) is pd.DataFrame):
    financial_data = pd.concat([financial_data, ticker.all_financial_data()])
    count += 1

```

- Dict to DataFrame

```

df_asset_profile = pd.DataFrame.from_dict(asset_profile)
df_long_business_summary = pd.DataFrame.from_dict(long_business_summary)
df_recommendations = pd.DataFrame.from_dict(recommendations)

```

- DataFrame to csv File

```

df_asset_profile.to_csv("C:/Users/SSAFY/Desktop/Project2/API_SCRIPT/asset_profile.csv")
df_long_business_summary.to_csv("C:/Users/SSAFY/Desktop/Project2/API_SCRIPT/long_business_summary.csv")
df_recommendations.to_csv("C:/Users/SSAFY/Desktop/Project2/API_SCRIPT/recommendations.csv")
grading_history.to_csv("C:/Users/SSAFY/Desktop/Project2/API_SCRIPT/grading_history.csv")
corporate_events.to_csv("C:/Users/SSAFY/Desktop/Project2/API_SCRIPT/corporate.csv")
financial_data.to_csv("C:/Users/SSAFY/Desktop/Project2/API_SCRIPT/financial_data.csv")

```

<https://velog.io/write?id=3f906e06-51ff-4913-acd1-a9131f155c0c>

네이버 파이낸셜 뉴스 수집

- ▼ 수집코드

```

import time
import pandas as pd
import re
import requests
import os
from bs4 import BeautifulSoup
from urllib.request import urlopen
from urllib.request import HTTPError
from urllib import parse
import numpy as np
import time
from tqdm import tqdm
from datetime import date, timedelta

```

```

from concurrent.futures import ThreadPoolExecutor
from fake_useragent import UserAgent

# 라이브러리를 설치하려면 아래 명령어를 실행하세요.
# pip install fake-useragent

ua = UserAgent()

def daterange(d1, d2):

    return [d1 + timedelta(days=i) for i in range((d2 - d1).days + 1)]

def request_with_retry(url, max_retries=3):
    headers = {'User-Agent': ua.random}
    for _ in range(max_retries):
        try:
            response = requests.get(url, headers=headers)
            return response
        except requests.exceptions.RequestException as e:
            print(f"Request failed: {e}, retrying...")
            time.sleep(1)
    raise Exception("Request failed after multiple retries")

def newItemCraw(search, stDataStart, stDataEnd):

    euc_data = search.encode('euc-kr')
    encodeCompany = str(euc_data).replace("\\x", "%")[2:-1]
    # https://finance.naver.com/news/news_search.naver?rcdate=&q=%bb%ef%bc%ba%c0%fc%c0%da&x=0&y=0&sm=all.basic&pd=1&stDateStart=19

    page = 1
    url = f"https://finance.naver.com/news/news_search.naver?rcdate=&q={encodeCompany}&x=0&y=0&sm=all.basic&pd=1&stDateStart={stDataStart}&page={page}"
    req = request_with_retry(url)
    bs = BeautifulSoup(req.content, "lxml", from_encoding='utf-8')
    newsTitleList = bs.find_all(class_='articleSubject')
    newsSummeryList = bs.find_all(class_='articleSummary')
    spanList = bs.find_all("span")
    res = {}

    for span in spanList:
        span.decompose()
    # summery = bs.find_all
    title_res = list()
    for news in newsTitleList:
        title_string = news.get_text() # save pd
        title_string = title_string.replace("\n", "")
        title_res.append(title_string)
        article_link = 'https://finance.naver.com' + \
            news.find('a')['href'] # save pd
        # 헤드 라인 추가
        # print(next)
        # print(title_string)
    if len(title_res) < 1:
        return None

    summery_res = list()
    for sumeryEL in newsSummeryList:

        sumeryString = sumeryEL.get_text()
        sumeryString = sumeryString.replace("\n", "")
        sumeryString = sumeryString.replace("\t", "")
        summery_res.append(sumeryString)
        # print(sumeryString)

    res["title"] = title_res
    res["summery"] = summery_res
    res["article_link"] = article_link

    return res

def newItemCrawJustOne(search, stDataStart, stDataEnd):

    euc_data = search.encode('euc-kr')
    encodeCompany = str(euc_data).replace("\\x", "%")[2:-1]
    # https://finance.naver.com/news/news_search.naver?rcdate=&q=%bb%ef%bc%ba%c0%fc%c0%da&x=0&y=0&sm=all.basic&pd=1&stDateStart=19

    page = 1
    url = f"https://finance.naver.com/news/news_search.naver?rcdate=&q={encodeCompany}&x=0&y=0&sm=all.basic&pd=1&stDateStart={stDataStart}&page={page}"
    req = requests.get(url)
    bs = BeautifulSoup(req.content, "lxml", from_encoding='utf-8')
    newsTitleList = bs.find(class_='articleSubject')
    newsSummeryList = bs.find(class_='articleSummary')
    spanList = bs.find_all("span")
    res = {}

```

```

for span in spanList:
    span.decompose()
# sumery = bs.find_all
title_res = list()
if newsTitleList == None:
    return None
title_string = newsTitleList.get_text() # save pd
title_string = title_string.replace("\n", "")
title_res.append(title_string)
article_link = 'https://finance.naver.com' + \
    newsTitleList.find('a')['href'] # save pd
# 헤드 라인 추가
# print(next)
# print(title_string)
if len(title_res) == 1:
    return None

sumery_res = list()

sumeryString = newsSumeryList.get_text()
sumeryString = newsSumeryList.replace("\n", "")
sumeryString = newsSumeryList.replace("\t", "")
sumery_res.append(sumeryString)
# print(sumeryString)

res["title"] = title_res
res["sumery"] = sumery_res
res["article_link"] = article_link

return res

def scrape_news(profile, year, find_date):
    day_news = newsItemCraw(profile, find_date, find_date)
    if day_news == None:
        return None
    else:
        day_news_df = pd.DataFrame({
            'ko_name': profile,
            'title': [day_news['title']],
            'summary': [day_news['sumery']],
            'article_link': [day_news['article_link']],
            'date': [find_date]
        })

        return day_news_df

def clean_text(text):
    if isinstance(text, list):
        return [t.encode('euc-kr', 'ignore').decode('euc-kr') for t in text]
    else:
        return text.encode('euc-kr', 'ignore').decode('euc-kr')

if __name__ == "__main__":
    get_cop_profile = pd.read_csv(
        "./asset_profile.csv", encoding='euc-kr')
    result = pd.DataFrame()
    ko_name_series = get_cop_profile["ko_name"]
    test = [y for y in range(2022, 2023)]
    year_set_1 = [y for y in range(1997, 2001+1)]#
    year_set_2 = [y for y in range(2002, 2006+1)]#
    year_set_3 = [y for y in range(2007, 2011+1)]
    year_set_4 = [y for y in range(2012, 2016+1)]
    year_set_5 = [y for y in range(2017, 2021+1)]
    year_set_6 = [y for y in range(2022, 2023+1)]
    covid = [y for y in range(2019, 2022+1)]#
    riemann = [y for y in range(2008, 2009+1)]
    dot_com = [y for y in range(1997, 2002+1)]
    ko_name_series = ko_name_series[439+186:]
    ko_name_series_1 = ko_name_series[:len(ko_name_series)//4] ## 완료
    ko_name_series_2 = ko_name_series[len(ko_name_series)//4:len(ko_name_series)//2] ## 완료
    ko_name_series_3 = ko_name_series[len(ko_name_series)//2:3*len(ko_name_series)//4] ## 완료
    ko_name_series_4 = ko_name_series[(3*len(ko_name_series)//4)-1:] ## 진행

    for profile in tqdm(ko_name_series_4, desc="기업 진행", position=2):
        for year in tqdm(covid, desc="년도 진행도", position=1):
            start_date = date(year, 1, 1)
            end_date = date(year, 12, 31)
            date_range = list(daterange(start_date, end_date))
            # Create a ThreadPoolExecutor and use it to execute scrape_news in parallel
            with ThreadPoolExecutor() as executor:
                tasks = [executor.submit(
                    scrape_news, profile, year, find_date) for find_date in date_range]
                scraped_news = [task.result() for task in tasks]

            # Filter out None values and update the result DataFrame

```

```

scraped_news = [news for news in scraped_news if news is not None]
if os.path.exists("covid.csv"):
    result = pd.read_csv("covid.csv", encoding='euc-kr')
else:
    result = pd.DataFrame()
# 기존 DataFrame에 새로운 데이터 추가하기
for news in scraped_news:
    result = pd.concat([result, news], ignore_index=True)
result['ko_name'] = result['ko_name'].apply(clean_text)
result['title'] = result['title'].apply(clean_text)
result['summary'] = result['summary'].apply(clean_text)
result['article_link'] = result['article_link'].apply(clean_text)
result.to_csv("covid.csv", encoding='euc-kr', index=False)

```

코드 개요

사용하는 테마 닷컴버블, 코로나, 리만브라더스 기간 에 맞춰 네이버 파이낸셜 뉴스 를 일별 10개를 스크래핑 하는 코드입니다
기간은

covid = 2019, 2022

riemann = 2008, 2009

dot_com = 1997, 2002

```

covid = [y for y in range(2019, 2022+1)]#
riemann = [y for y in range(2008, 2009+1)]
dot_com = [y for y in range(1997, 2002+1)]

```

으로구성 되어 있습니다

한기업에 대해 테마 년도 365일을 순회 하며 365를 순회할땐 멀티쓰래드를 구성해 수집 속도를 개선 했습니다.

```

with ThreadPoolExecutor() as executor:
    tasks = [executor.submit(
        scrape_news, profile, year, find_date) for find_date in date_range]
    scraped_news = [task.result() for task in tasks]

```

수집 방법

하드웨어 성능에 따라 다르지만 메모리 점유를 오래 할시 수집 속도가 급격히 내려가는걸 확인했고

이를 해결하기 위해

다음을 수동으로 끝나면 바뀌가며 수행해야합니다.

```

for profile in tqdm(ko_name_series_4, desc="기업 진행", position=2):
    for year in tqdm(covid, desc="년도 진행도", position=1):

```