한국 3대 포털 기사 WordCloud 분석 자동화 시스템

고 건 호

Project Background & Goal

기획배경

정보의 홍수

너무 많이 배출되는 뉴스의 양

뉴스보다 SNS 선호하는 젊은 세대

뉴스 볼 여유가 없는 바쁜 직장인들

꾸준히 읽지 않으면 알기 힘든 트렌드

기획목표

매일 정해진 시간에 자동 실행

한국 3대 검색엔진의 기사 수집

한눈에 볼 수 있도록 WordCloud 분석

편의성 증대를 위해 Slack Messenger로 전송

Project Overview

개요

1. 한국 3대 포털엔진 기사 수집 - Naver, Daum, Nate
2. Scrapy의 Pipeline 기능 활용 - MongoDB 저장
3. MongoDB에 저장된 데이터 취합 (toWordCloud.py)
4. WordCloud 분석 (toWordCloud.py)
5. AWS CLI 명령어로 AWS S3에 WordCloud 이미지 저장
6. Image URL 추출 후, Slack Messenger로 전송

- 위 모든 작업은 crontab 기능으로 시간 지정하여 실행

Skills

사용언어 : Python

자료수집 : Scrapy, Xpath

데이터베이스 : MongoDB, AWS S3

언어분석 : konlpy, nltk

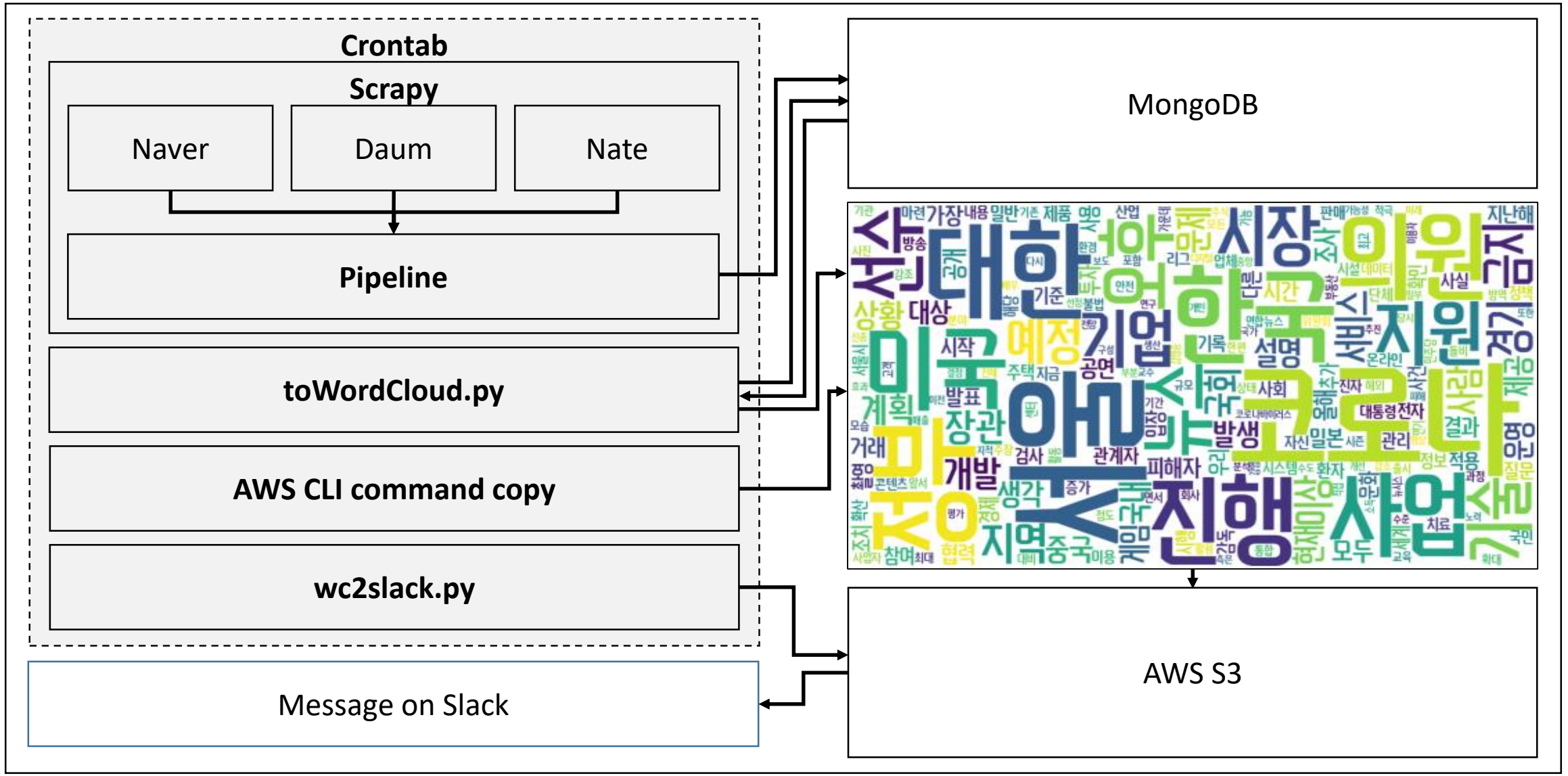
시각화 : WordCloud

자동화 : Crontab

서버 : AWS EC2

메신저 : Slack

Project Overview



Project Detail - 1. 기사 수집

Scrapy

1. 기사 수집 용 Scrapy 프로젝트 3개 생성
2. 파이프라인으로 수집된 데이터 MongoDB 저장

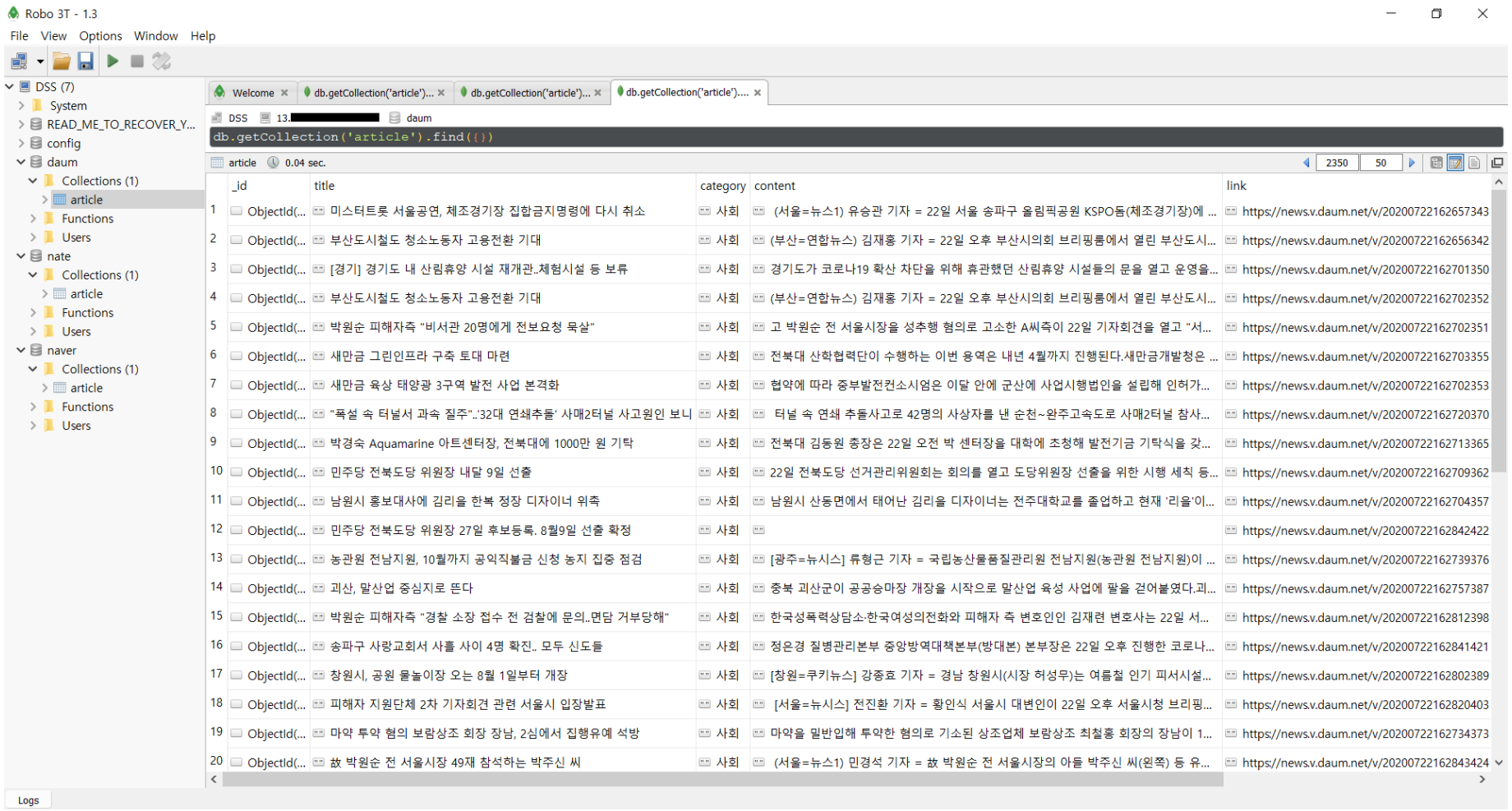
```python
import scrapy
from newsDaum.items import NewsdaumItem
# from selenium import webdriver
from scrapy.http import TextResponse
import requests

class DaumSpider(scrapy.Spider):
    name = "NewsDaum"
    allow_domain = ["https://daum.net"]
    start_urls = ["https://news.daum.net/breakingnews/economic"]

    def parse(self, response):
        categories = ['society', 'politics', 'economic', 'foreign', 'culture', 'entertain', 'sports', 'digital']
        for name in categories:
            for page in range(1, 100):
                url = "https://news.daum.net/breakingnews/{}?page={}".format(name, page)
                req = requests.get(url)
                response = TextResponse(req.url, body=req.text, encoding="utf-8")
                links  = response.xpath('//*[@id="mArticle"]/div[3]/ul/li/div/strong/a/@href').extract()
                for link in links:
                    yield scrapy.Request(link, callback=self.parse_content)

    def parse_content(self, response):
        item = NewsdaumItem()
        item['title'] = response.xpath('//*[@id="cSub"]/div/h3')[0].extract().split(">")[1].split("<")[0]
        item['category'] = response.xpath('//*[@id="kakaoBody"]')[0].extract().split(">")[1].split("<")[0]
        content = response.xpath('//*[@id="harmonyContainer"]/section/p/text()').extract()
        item['content'] = "".join(content)
        item['link'] = response.url
        yield item
```

```python
from itemadapter import ItemAdapter
from .mongodb import collection

class NewsdaumPipeline:

    def process_item(self, item, spider):

        data = {
            "title": item["title"],
            "category": item["category"],
            "content": item["content"],
            "link": item["link"],
        }
        collection.insert(data)

        return item
```

Project Detail - 2. 데이터 불러오기

MongoDB

데이터베이스에 저장된 데이터 취합 (toWordCloud.py)



Project Detail - 3. 언어 분석

Python

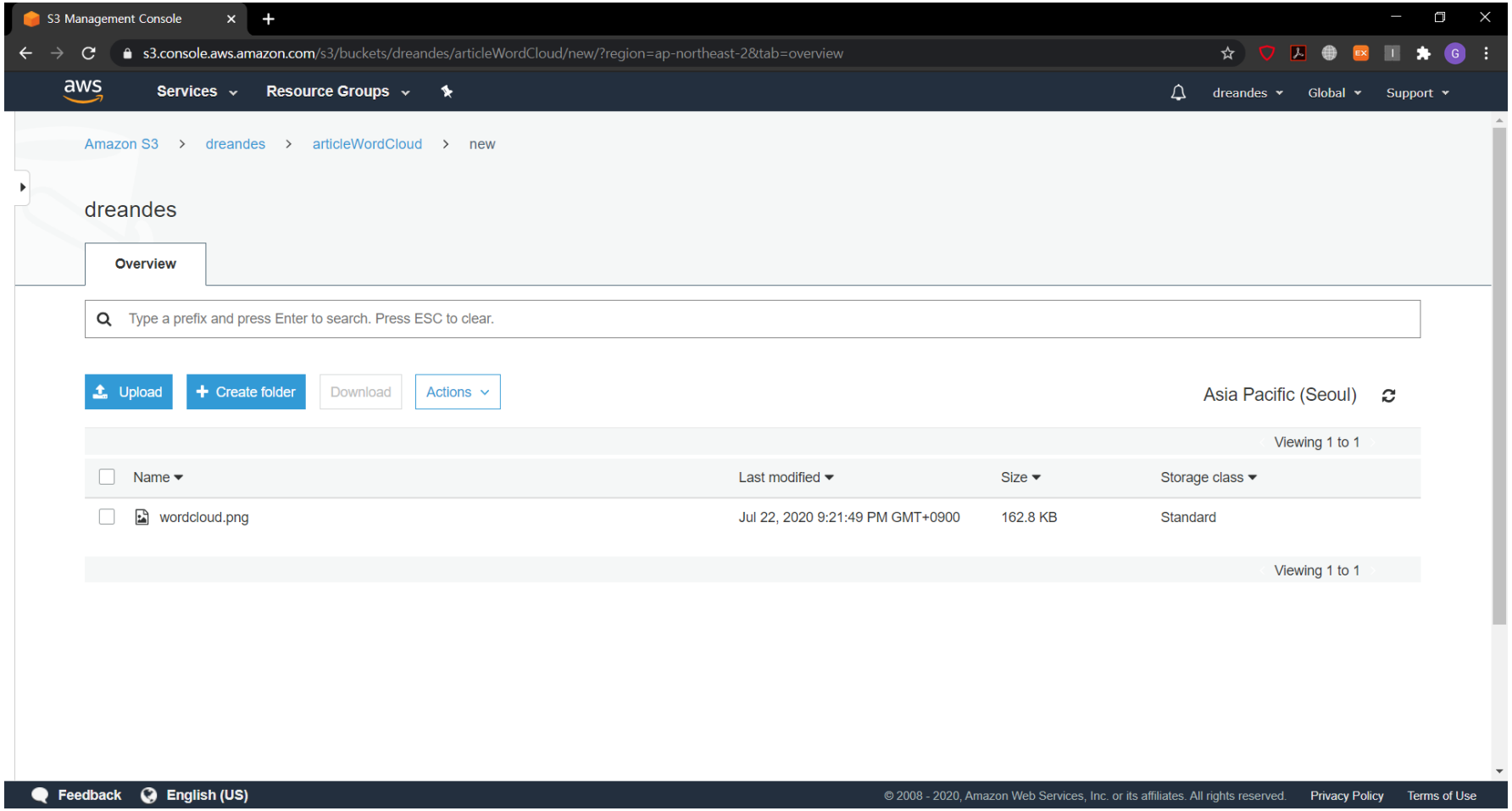
1. konlpy 패키지 활용하여 명사 추출
2. nltk 패키지 활용하여 추출된 명사 분석
3. WordCloud 이미지 추출 및 서버 저장


```python
from konlpy.tag import Okt
from tqdm import tqdm
from nltk import FreqDist
from wordcloud import WordCloud
from datetime import datetime
import pandas as pd
import matplotlib.pylab as plt

import pymongo

import requests, json


client = pymongo.MongoClient("mongodb://13.                ")
result_nate = client['nate'].article
result_daum = client['daum'].article
result_naver = client['naver'].article

df_nate = pd.DataFrame(list(result_nate.find()))
df_daum = pd.DataFrame(list(result_daum.find()))
df_naver = pd.DataFrame(list(result_naver.find()))

df = df_nate.append(df_daum)
df = df.append(df_naver).reset_index()

df = df.drop(['index', '_id'], axis=1)


def tokenize(doc):
    tagger = Okt()
    tokens = [t for t in tagger.nouns(doc)]
    return tokens

def towordcloud(df):
    df = df.dropna()
    docs = tuple([x for x in df.to_numpy()])
    sentences = []
    for d in tqdm(docs):
        tokens = [token for token in tokenize(d) if token.isalnum()]
        sentences.append(tokens)
    words = [word for sentence in sentences for word in sentence]
    words = [word for word in words if len(word) > 1]
    words_remove = ['으로', '에서', '에도', '했다', '있다', '이다', '무단', '배포', '위해', '대표', '때문',
                    '그룹', '통해', '최근', '경우', '이번', '이후', '라며', '지난', '대해', '기자', '관련',
                   ]
    words_r = [word for word in words if word not in words_remove]
    fd = FreqDist(words_r)
#     print(fd.most_common(20))
    font_path = '/home/ubuntu/python3/Crawling/koverwatch.ttf'
    wc = WordCloud(width=1000, height=600, background_color="white", random_state=0,
                   font_path=font_path)
    plt.imshow(wc.generate_from_frequencies(fd))
    plt.axis("off")
    str = "/home/ubuntu/python3/Crawling/wordcloud_" + datetime.now().strftime("%Y.%m.%d_%H.%M.%S") + ".png"
#    plt.savefig(str)
    plt.savefig('/home/ubuntu/python3/Crawling/wordcloud.png')
#    plt.show()

towordcloud(df['content'])
```

Project Detail - 4. Amazon S3 저장

AWS CLI

AWS CLI 명령어를 통해 wordcloud.png 서버 복사

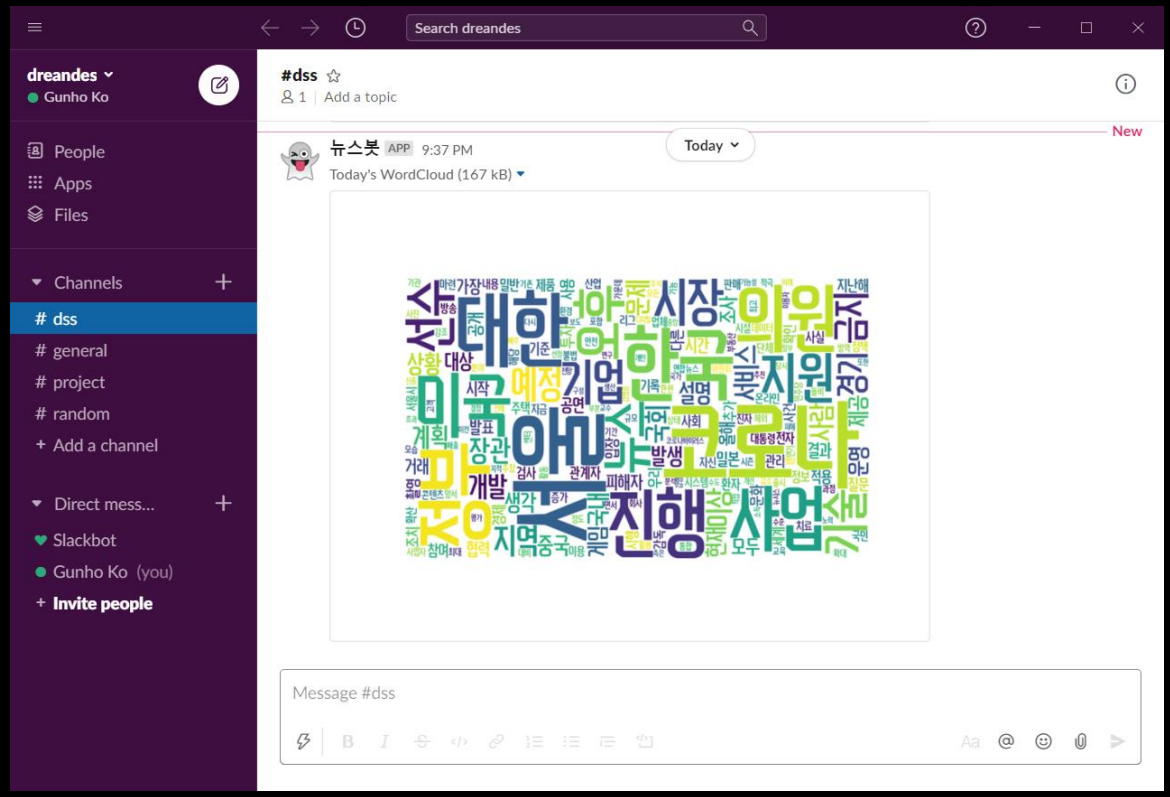


Project Detail - 5. image_url 추출 후, Slack 전송

Slack

image_url 추출 후, Slack Messenger로 전송

```
import requests, json
import boto3

s3 = boto3.client('s3')
image_url = s3.generate_presigned_url('get_object',
                        Params={'Bucket': 'dreandes',
                                'Key':'articleWordCloud/new/wordcloud.png'})

def send_msg(slack_webhook, block, channel="#dss", username="뉴 스 봇 "):
    payload = {"channel": channel,
               "username": username,
#               "text": msg,
               "icon_emoji": ":ghost:",
               "blocks": block
              }
    requests.post(slack_webhook, json.dumps(payload))

slack_webhook = "https://hooks.slack.com/services/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"

block = [{"type": "image",
          "title": {
              "type": "plain_text","text": "Today's WordCloud","emoji": True
          },
          "image_url": image_url,
          "alt_text": "wordcloud"}]


#msg = "Today's WordCloud"

send_msg(slack_webhook, block)
```



Project Detail - 6. 자동화

Crontab

Crontab 실행 시간 지정을 통한 자동화 시스템 구축

ubuntu@ip-[redacted]: ~

```
PATH=/home/ubuntu/.pyenv/versions/python3/bin/

# Scrapy
30 16 * * * cd /home/ubuntu/python3/Crawling/newsNate/newsNate && scrapy crawl NewsNate
30 16 * * * cd /home/ubuntu/python3/Crawling/newsDaum/newsDaum && scrapy crawl NewsDaum
30 16 * * * cd /home/ubuntu/python3/Crawling/newsNaver/newsNaver && scrapy crawl NewsNaver

# Get WordCloud
34 16 * * * cd /home/ubuntu/python3/Crawling && python toWordCloud.py

# Copy to S3
39 16 * * * /usr/bin/aws s3 cp /home/ubuntu/python3/Crawling/wordcloud.png s3://dreandes/articleWordCloud/new/

# Send WordCloud to Slack
40 16 * * * cd /home/ubuntu/python3/Crawling && python wc2slack.py

# Move past WordClouds to old directory
45 16 * * * /usr/bin/aws s3 mv s3://dreandes/articleWordCloud/new s3://dreandes/articleWordCloud/old --recursive
45 16 * * * /bin/mv /home/ubuntu/python3/Crawling/wordcloud.png /home/ubuntu/python3/Crawling/wordcloud/


# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                    1,1           All
```

감사합니다.