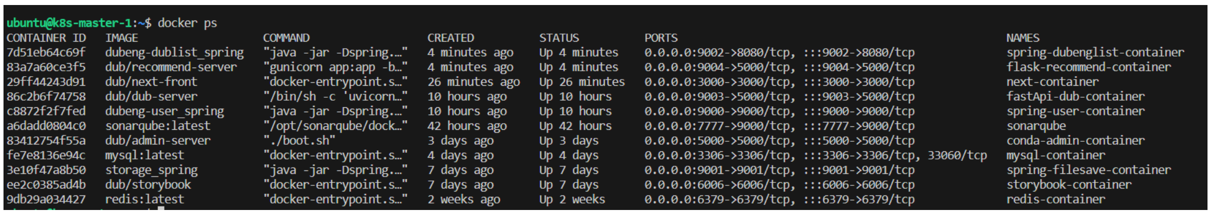
Dubeng 덥잉 - 포팅 메뉴얼

[테스트 환경] - 단일 EC2

[도메인] : k8b208.p.ssafy.io

Docker 컨테이너 리스트



```
ubuntu@k8s-master-1:~$ docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
7d51eb64c69f   dubeng-dublist_spring     "java -jar -Dspring.…"   4 minutes ago    Up 4 minutes    0.0.0.0:9002->8080/tcp, :::9002->8080/tcp              spring-dubenglist-container
83a7a60ce3f5   dub/recommend-server      "gunicorn app:app -b…"   4 minutes ago    Up 4 minutes    0.0.0.0:9004->5000/tcp, :::9004->5000/tcp              flask-recommend-container
29ff44243d91   dub/next-front            "docker-entrypoint.s…"   26 minutes ago   Up 26 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp              next-container
86c2b6f74758   dub/dub-server            "/bin/sh -c 'uvicorn…"   10 hours ago     Up 10 hours     0.0.0.0:9003->5000/tcp, :::9003->5000/tcp              fastApi-dub-container
c8872f2f7fed   dubeng-user_spring        "java -jar -Dspring.…"   10 hours ago     Up 10 hours     0.0.0.0:9000->9000/tcp, :::9000->9000/tcp              spring-user-container
a6dadd0804c0   sonarqube:latest          "/opt/sonarqube/dock…"   42 hours ago     Up 42 hours     0.0.0.0:7777->9000/tcp, :::7777->9000/tcp              sonarqube
83412754f55a   dub/admin-server          "./boot.sh"              3 days ago       Up 3 days       0.0.0.0:5000->5000/tcp, :::5000->5000/tcp              conda-admin-container
fe7e8136e94c   mysql:latest              "docker-entrypoint.s…"   4 days ago       Up 4 days       0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql-container
3e10f47a8b50   storage_spring            "java -jar -Dspring.…"   7 days ago       Up 7 days       0.0.0.0:9001->9001/tcp, :::9001->9001/tcp              spring-filesave-container
ee2c0385ad4b   dub/storybook             "docker-entrypoint.s…"   7 days ago       Up 7 days       0.0.0.0:6006->6006/tcp, :::6006->6006/tcp              storybook-container
9db29a034427   redis:latest              "docker-entrypoint.s…"   2 weeks ago      Up 2 weeks      0.0.0.0:6379->6379/tcp, :::6379->6379/tcp              redis-container
```

▼ Docker 설치

```
sudo apt-get update -y
sudo apt-get install     ca-certificates     curl     gnupg     lsb-release

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo   "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io -y

sudo usermod -aG docker $USER

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bi

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version
```

▼ letsencrypt SSL 인증서 발급

Let’s Encrypt

> SSL 인증서 발급자(인증 기관 또는 CA라고 함) 중 하나로서 무료로 SSL 인증서를 발급해준다. 다만 무료이니 만큼 인증서 유효기간은 90일이기 때문에 3개월에 한번씩 인증서 갱신을 해주어야 한다.

🎇 자동으로 갱신해주는 패키지도 있으니 찾아보도록 하자

1. **EC2에 Let’s Encrypt 설치**

```
sudo apt-get install letsencrypt
```

2. **인증서 적용 및 pem Key 얻기**

```
sudo letsencrypt certonly --standalone -d [도메인]
```

- 이메일 입력 (선택사항)
- 서비스 이용 동의 (필수)
- 정보 수집 (선택 사항)

이렇게 인증서를 성공적으로 발급 받으면 아래 경로에 key가 발급받은 것을 볼 수 있다.

```
sudo ls -al /etc/letsencrypt/live/[도메인]

ubuntu@ip-172-26-12-167:~$ sudo ls -al /etc/letsencrypt/live/k8b208.p.ssafy.io
-rw-r--r-- 1 root root  692 Feb  9 17:54 README
lrwxrwxrwx 1 root root   41 Feb  9 17:54 cert.pem -> ../../archive/k8b208.p.ssafy.io/cert1.pem
lrwxrwxrwx 1 root root   42 Feb  9 17:54 chain.pem -> ../../archive/k8b208.p.ssafy.io/chain1.pem
lrwxrwxrwx 1 root root   46 Feb  9 17:54 fullchain.pem -> ../../archive/k8b208.p.ssafy.io/fullchain1.pem
-rw------- 1 root root 5786 Feb 15 23:40 keystore.p12
lrwxrwxrwx 1 root root   44 Feb  9 17:54 privkey.pem -> ../../archive/k8b208.p.ssafy.io/privkey1.pem
```

3. **Nginx 설정파일을 아래와 같이 수정한다. (/etc/nginx/conf.d/default.conf)**

```
// 서버 80포트로 들어오는 요청을 https로 리다이렉트 시켜준다.
server {
        listen 80;
        server_name k8b208.p.ssafy.io;
        return 301 https://$server_name$request_uri;
}
// 서버 443 포트로 들어오는 요청에 SSL 인증서를 적용한다.
// /etc/letsencrypt/ 는 nginx container -v 옵션으로 볼륨을 주어 연결을 하였다.

server {
    listen 443 ssl;
    server_name k8b208.p.ssafy.io

    ssl_certificate /etc/letsencrypt/live/k8b208.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k8b208.p.ssafy.io/privkey.pem;
}
```

▼ **Nginx 설치**

Ubuntu EC2 에 Nginx 설치

- docker를 통한 nginx 설치
- EC2 인스턴스 내에 자체적으로 nginx 설치

설치, 설정의 편의성을 위해 EC2 내에 자체적으로 Nginx 설치를 하도록하겠다.

```
sudo apt-get update
sudo apt-get install nginx
```

nginx conf 파일에 client_max_body_size 설정

nginx의 기본 업로드 제한이 1MB이기 때문에 발생하는 문제이다.

해당 사이즈를 원하는 크기로 변경하면 정상적으로 파일이 업로드가 된다.

```
vi /etc/nginx/nginx.conf

http {
        ##
        # Basic Settings
        ##
        client_max_body_size 10M;
}
```

default.conf

sudo vi /etc/nginx/conf.d/default.conf

```
upstream jenkins {
        keepalive 32; #keepalive connections
        server localhost:8080;
}
# Required for Jenkins websocket agents
map $http_upgrade $connection_upgrade {
  default upgrade;
  '' close;
}

server {
        listen 443 ssl;
        server_name k8b208.p.ssafy.io;

        # SSL Certificate
        ssl_certificate /etc/letsencrypt/live/k8b208.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/k8b208.p.ssafy.io/privkey.pem;


        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                #try_files $uri $uri/ =404;
                #root /var/www/html;
                #index index.html index.htm index.nginx-debian.html;
                proxy_pass http://localhost:3000;

                proxy_redirect     default;
                proxy_http_version 1.1;

                # Required for Jenkins websocket agents
                proxy_set_header   Connection        $connection_upgrade;
                proxy_set_header   Upgrade           $http_upgrade;

                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;
                proxy_max_temp_file_size 0;
        }
        location /record {
                proxy_pass http://localhost:9003;

                proxy_redirect     default;
                proxy_http_version 1.1;

                # Required for Jenkins websocket agents
                proxy_set_header   Connection        $connection_upgrade;
                proxy_set_header   Upgrade           $http_upgrade;

                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;
                proxy_max_temp_file_size 0;
        }
        location /recommend {
                proxy_pass http://localhost:9004;
        }
        location /file {
                proxy_pass http://localhost:9001;

                proxy_redirect     default;
                proxy_http_version 1.1;

                # Required for Jenkins websocket agents
                proxy_set_header   Connection        $connection_upgrade;
                proxy_set_header   Upgrade           $http_upgrade;

                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;
                proxy_max_temp_file_size 0;

                add_header 'Access-Control-Allow-Origin' '*' always;
                add_header 'Access-Control-Allow-Methods' '*';
        }
        location /storybook {
                proxy_pass http://localhost:6006;
        }
        location /user {
                proxy_pass http://localhost:9000;
        }
        location /jenkins {
                proxy_pass http://localhost:8080;
                proxy_redirect     default;
                proxy_http_version 1.1;

                # Required for Jenkins websocket agents
                proxy_set_header   Connection        $connection_upgrade;
                proxy_set_header   Upgrade           $http_upgrade;

                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;
                proxy_max_temp_file_size 0;
        }
        location /admin {
                proxy_pass http://localhost:5000;

                add_header 'Access-Control-Allow-Origin' '*' always;
                add_header 'Access-Control-Allow-Methods' '*';
        }
        location /dub {
                proxy_pass http://localhost:9002;
        }
}
server {
        listen 80;
        listen [::]:80;

        server_name k8b208.p.ssafy.io;

        return 301 https://$server_name$request_uri;
}
```

▼ Jenkins 설정

▼ Spring - FileServer

[Dockerfile]

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar

EXPOSE 9001

COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

[**docker-compose.yml]**

```
version: '3'
services:
  spring:
    container_name: 'spring-filesave-container'
    build:
      context: .    # Build Context Directory
      dockerfile: ./Dockerfile      # Dockerfile
    ports:
      - "9001:9001"
    volumes:    # Connect Local Volume
      - /home/ubuntu/file_volume:/Home
```

[Jenkins] pipeline Script

```
pipeline {
    agent any
    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/filesave',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SPRING BUILD'){
            steps{
                dir('back/storage'){
                    sh '''
                    cp /home/ubuntu/env/storage_server/application-dev.yml ./src/main/resources/application-dev.yml
                    chmod +x ./gradlew
                    ./gradlew clean build -x test
                    '''
                }
            }
        }
        stage('DEPLOY'){
            steps{
                dir('back/storage'){
                    sh '''
                        ls -al
                        docker-compose down || true
                        docker-compose up -d --build
                    '''
                }
            }
        }
    }
}
```

▼ Jenkins Pipeline - docker-compose

```
pipeline {
    agent any
    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'feature-back/add-spring-filesave',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SPRING BUILD'){
            steps{
                dir('back/storage'){
                    sh '''
                    chmod +x ./gradlew
                    ./gradlew clean build
                    '''
                }
            }
        }
        stage('DOCKER BUILD'){
            steps{
                dir('back/storage'){
                    sh '''
                        ls -al
                        docker-compose down || true
                        docker-compose up -d
                    '''
                }
            }
        }
    }
}
```

▼ **Spring - dublist Server**

[Docker] Dockerfile

```
FROM openjdk:11-jdk

ARG JAR_FILE=build/libs/*.jar

COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","-Dspring.profiles.active=dev","/app.jar"]
```

[docker-compose.yml]

```
version: '3'
services:
  spring:
    container_name: 'spring-dubenglist-container'
    build:
      context: .    # Build Context Directory
      dockerfile: ./Dockerfile      # Dockerfile
    ports:
      - "9002:8080"
    volumes:    # Connect Local Volume
      - /home/ubuntu/file_volume:/Home
```

[Jenkins] Pipeline script

```
pipeline {
    agent any
    environment {
        Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
    }
    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/dubeng',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('BUILD'){
            steps{
                dir('back/dubeng-dublist'){
                    sh '''
                    cp /home/ubuntu/env/dublist_server/application-dev.yml ./src/main/resources/application-dev.yml
                    chmod +x ./gradlew
                    ./gradlew clean build -x test
                    '''
                }
            }
        }
        stage('SPRING-DEPLOY'){
            steps{
                dir('back/dubeng-dublist'){
                    sh '''
                        ls -al
                        docker-compose down || true
                        docker-compose up -d --build
                    '''
                }
            }
        }
    }
    post {
        success {
            mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
        failure {
            mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
    }
}
```

▼ **Spring - User Server**

[Docker] Dockerfile

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar

EXPOSE 9000

COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","-Dspring.profiles.active=dev","/app.jar"]
```

[docker-compose.yml]

```
version: '3'
services:
  spring:
    container_name: 'spring-user-container'
    build:
      context: .    # Build Context Directory
      dockerfile: ./Dockerfile      # Dockerfile
    ports:
      - "9000:9000"
```

[Jenkins] Pipeline script

```
pipeline {
    agent any

    environment {
        Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
    }

    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/user',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SPRING BUILD'){
            steps{
                dir('back/dubeng-user'){
                    sh '''
                    cp /home/ubuntu/env/user_server/application-dev.yml ./src/main/resources/application-dev.yml
                    chmod +x ./gradlew
                    ./gradlew clean build -x test
                    '''
                }
            }
        }
        stage('DEPLOY'){
            steps{
                dir('back/dubeng-user'){
                    sh '''
                        ls -al
                        docker-compose down || true
                        docker-compose up -d --build
                    '''
                }
            }
        }
    }
    post {
        success {
            mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
        failure {
            mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
    }
}
```

▼ **conda - Admin**

docker로 conda 가상환경을 구축하고 ffepmg를 설치하고 fastAPI를 docker image화 시킨다.

[Dockerfile]

```
FROM continuumio/miniconda:latest

WORKDIR /app

COPY . .

RUN chmod +x boot.sh

RUN conda env create -f environment.yml

RUN echo "source activate admin-environment" >> ~/.bashrc
ENV PATH /opt/conda/envs/admin-environment/bin:$PATH

RUN apt-get --allow-releaseinfo-change update
RUN apt-get install -y ffmpeg

RUN pip install -r requirements.txt

EXPOSE 5000

ENTRYPOINT ["./boot.sh"]
```

[environment.yml]

```
name: admin-environment
channels:
- defaults
dependencies:
- python=3.8
- flask
- gunicorn
```

boot.sh

실행에 필요한 쉘 스크립트 파일 (gunicorn 설정)

```
#!/bin/sh
# -t 240 : timeout 설정
exec gunicorn -b :5000 --access-logfile - -t 240 --error-logfile - app:app
```

requirements.txt

pipeline 모듈 설치를 위한 config 파일

```
pip freeze > requirements.txt
```

▼ **requirements.txt**

```
absl-py==1.4.0
anyio==3.6.2
asttokens==2.2.1
astunparse==1.6.3
audioread==3.0.0
backcall==0.2.0
blinker==1.6.2
boto3==1.26.121
botocore==1.29.121
cachetools==5.3.0
certifi==2022.12.7
cffi==1.15.1
charset-normalizer==3.1.0
click==7.1.2
colorama==0.4.6
decorator==5.1.1
executing==1.2.0
ffmpeg-python==0.2.0
Flask==2.0.0
flatbuffers==23.3.3
future==0.18.3
gast==0.4.0
google-api-core==2.11.0
google-api-python-client==2.86.0
google-auth==2.17.3
google-auth-httplib2==0.1.0
google-auth-oauthlib==1.0.0
google-pasta==0.2.0
googleapis-common-protos==1.59.0
grpcio==1.54.0
h11==0.12.0
h2==4.1.0
h5py==3.8.0
hpack==4.0.0
httpcore==0.13.7
httplib2==0.22.0
httpx==0.19.0
hyperframe==6.0.1
idna==3.4
importlib-metadata==6.6.0
ipython==8.12.0
itsdangerous==2.1.2
jax==0.4.8
jedi==0.18.2
Jinja2==3.1.2
jmespath==1.0.1
joblib==1.2.0
keras==2.12.0
libclang==16.0.0
librosa==0.8.1
llvmlite==0.38.1
Markdown==3.4.3
MarkupSafe==2.1.2
matplotlib-inline==0.1.6
ml-dtypes==0.1.0
norbert==0.2.1
numba==0.55.2
numpy==1.22.4
oauthlib==3.2.2
opt-einsum==3.3.0
packaging==23.1
pandas==1.5.3
parso==0.8.3
pickleshare==0.7.5
platformdirs==3.2.0
pooch==1.7.0
prompt-toolkit==3.0.38
protobuf==3.20.3
pure-eval==0.2.2
pyasn1==0.5.0
pyasn1-modules==0.3.0
pycparser==2.21
pydub==0.25.1
Pygments==2.15.1
```

```
PyMySQL==1.0.3
pyparsing==3.0.9
python-dateutil==2.8.2
pytube==12.1.3
pytz==2023.3
requests==2.28.2
requests-oauthlib==1.3.1
resampy==0.4.2
rfc3986==1.5.0
rsa==4.9
s3transfer==0.6.0
scikit-learn==1.2.2
scipy==1.10.1
six==1.16.0
sniffio==1.3.0
soundfile==0.12.1
spleeter==2.3.2
stack-data==0.6.2
termcolor==2.2.0
threadpoolctl==3.1.0
traitlets==5.9.0
typer==0.3.2
typing_extensions==4.5.0
uritemplate==4.1.1
urllib3==1.26.15
waitress==2.1.2
wcwidth==0.2.6
Werkzeug==2.3.1
wrapt==1.14.1
xmltodict==0.13.0
youtube-transcript-api==0.6.0
zipp==3.15.0
```

python env 환경 변수

```
#환경 변수 파일은 아래의 경로에 위치한다.
/home/ubuntu/env/admin_server/env-vedioInfo.txt
/home/ubuntu/env/admin_server/env.txt

# Jenkins 실행 시, 이미지 빌드 전 환경변수 파일을 import 해준다.
cp /home/ubuntu/env/admin_server/env-vedioInfo.txt /var/lib/jenkins/workspace/dub-admin-server/back/dubeng-admin/env-vedioInf
cp /home/ubuntu/env/admin_server/env.txt /var/lib/jenkins/workspace/dub-admin-server/back/dubeng-admin/env.txt
```

Jenkins Script

```
pipeline{
    agent any
    // 환경변수 세팅
    environment {
        Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
    }

    stages{
        stage('GIT CLONE'){
            steps{
                git branch : 'develop-back/admin',
                        credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SETTING ENV'){
            steps{
                dir('back/dubeng-admin'){
                    sh '''
                    cp /home/ubuntu/env/admin_server/env-vedioInfo.txt ./env-vedioInfo.txt
                    cp /home/ubuntu/env/admin_server/env.txt ./env.txt
                    '''
                }
            }
        }
        stage('DOCKER BUILD'){
            steps{
                dir('back/dubeng-admin'){
                    sh '''
                        docker stop conda-admin-container || true
                        docker rm conda-admin-container || true
                        docker rmi dub/admin-server || true

                        docker build -t dub/admin-server .
                    '''
                }
            }
        }
        stage('DEPLOY'){
            steps{
                sh '''
                    docker run --name conda-admin-container -p 5000:5000 -v /home/ubuntu/admin_storage:/download/dwn -d dub/a
                '''
                echo 'DEPLOY Success'
            }
        }
    }
    // end
    post {
        success {
            mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/[crediential]',
            channel: 'b208-jenkins-notification'
            )
        }
        failure {
            mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/[crediential]',
            channel: 'b208-jenkins-notification'
            )
        }
    }
}
```

▼ **FastAPI - dubeng Server**

[Docker] Dockerfile

```
# 베이스가 되는 Docker Image로 python 이미지를 사용
FROM python:3.8-slim

# 처음 실행 시 사용 되는 경로 정보 입니다.
WORKDIR /app

# 현재 경로의 main.py 및 모든 파일을 /app 경로로 복사합니다.
COPY . /app

# 현재 경로의 requirements.txt를 /app 경로로 복사합니다.
COPY requirements.txt /app

# 복사 된 requirements.txt를 사용하여 pip로 패키지를 추가합니다.
RUN pip install -r requirements.txt

# ffmpeg 설치
RUN apt-get --allow-releaseinfo-change update

RUN apt-get install -y ffmpeg

#  uvicorn을 사용하여 main.py의 app을 실행시킵니다.
CMD uvicorn --host=0.0.0.0 --port 5000 main:app
```

[requirements.txt]

```
anyio==3.6.2
blinker==1.6.2
boto3==1.26.127
botocore==1.29.127
certifi==2022.12.7
cffi==1.15.1
charset-normalizer==3.1.0
click==8.1.3
colorama==0.4.6
```

```
cryptography==40.0.2
fastapi==0.95.1
h11==0.14.0
idna==3.4
itsdangerous==2.1.2
Jinja2==3.1.2
jmespath==1.0.1
MarkupSafe==2.1.2
pycparser==2.21
pydantic==1.10.7
pydub==0.25.1
PyMySQL==1.0.3
python-dateutil==2.8.2
requests==2.30.0
s3transfer==0.6.1
six==1.16.0
sniffio==1.3.0
starlette==0.26.1
typing_extensions==4.5.0
urllib3==1.26.15
uvicorn==0.22.0
Werkzeug==2.3.3
```

[Jenkins] pipeline Script

```
pipeline {
    agent any

    environment {
        Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
    }

    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/user',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('FASTAPI BUILD'){
            steps{
                dir('back/dubeng-dub'){
                    sh '''
                        cp /home/ubuntu/env/dub_server/env.txt env.txt

                        docker rmi dub/dub-server || true
                        docker build -t dub/dub-server .
                    '''
                }
            }
        }
        stage('DEPLOY'){
            steps{
                dir('back/dubeng-dub'){
                    sh '''
                        docker stop fastApi-dub-container || true
                        docker rm fastApi-dub-container || true

                        docker run --name fastApi-dub-container -v /home/ubuntu/file_volume:/Home -p 9003:5000 -d dub/dub-serv
                    '''
                }
            }
        }
    }
    post {
        success {
            mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
        failure {
            mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
    }
}
```

▼ **Flask- recommend Server**

[Docker] Dockerfile

```
FROM python:3.8-slim

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

RUN pip install gunicorn

COPY . /app

WORKDIR /app

CMD ["gunicorn", "app:app", "-b", "0.0.0.0:5000", "--timeout", "300"]
```

requirements.txt

```
boto3==1.26.120
botocore==1.29.120
click==8.1.3
colorama==0.4.6
Flask==2.2.3
Flask-Cors==3.0.10
importlib-metadata==6.6.0
itsdangerous==2.1.2
Jinja2==3.1.2
jmespath==1.0.1
joblib==1.2.0
keras==2.11.0
MarkupSafe==2.1.2
numpy==1.24.1
opencv-python==4.7.0.68
pandas==2.0.1
pydub==0.25.1
PyMySQL==1.0.3
python-dateutil==2.8.2
pytz==2023.3
s3transfer==0.6.0
scikit-learn==1.2.2
scipy==1.10.1
six==1.16.0
threadpoolctl==3.1.0
tzdata==2023.3
urllib3==1.26.15
Werkzeug==2.2.3
zipp==3.15.0
```

[Jenkins] pipeline Script

```
pipeline {
    agent any
    environment {
        Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
    }
    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/dubeng',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('BUILD'){
            steps{
                dir('back/dubeng-recommend'){
                    sh '''
                    cp /home/ubuntu/env/recommend_server/env.txt ./env.txt
                    docker stop flask-recommend-container || true

                    docker rm flask-recommend-container || true

                    docker rmi dub/recommend-server || true

                    docker build -t dub/recommend-server .
                    '''
                }
            }
        }
        stage('FLASK DEPLOY'){
            steps{
                dir('back/dubeng-recommend'){
                    sh '''
                        docker run --name flask-recommend-container -p 9004:5000 -d dub/recommend-server
                    '''
                }
            }
        }
    }
    post {
        success {
            mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
        failure {
            mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
    }
}
```

▼ **Front - NextJS**

[Docker] Dockerfile

```
FROM node:16-alpine AS build

WORKDIR /app

COPY ./package.json /app

RUN npm install

# 어떤 파일이 이미지에 들어가야 하는지
# 첫 번째 .은 이 프로젝트의 모든 폴더 및 파일들 (Dockerfile을 제외한)
# 두 번째 .은 파일을 저장할 컨테이너 내부 경로 (ex /app)
COPY ./ /app

EXPOSE 3000

RUN npm run build

CMD ["npm", "run", "start"]
```

[Jenkins] Pipeline Script

```
pipeline{
    agent any

    environment {
        Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
    }

    stages{
        stage('GIT CLONE'){
            steps{
                git branch : 'develop-front',
                        credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SETTING ENV'){
            steps{
                dir('dubeng-front'){
                    sh '''
                        cp /home/ubuntu/env/front_server/.env .env
                    '''
                }
            }
        }
        stage('DOCKER BUILD'){
            steps{
                dir('dubeng-front'){
                    sh '''
                        docker build -t dub/next-front -f Dockerfile-next .
                    '''
                }
            }
        }
        stage('DEPLOY'){
            steps{
                sh '''
                    docker stop next-container || true
                    docker rm next-container || true
                    docker run --name next-container -d -p 3000:3000 dub/next-front
                '''
            }
        }
    }
    // end
    post {
        success {
            mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
        failure {
            mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
    }
}
```

▼ **Front - Storybook**

[Docker] Dockerfile

```
FROM node:16-alpine AS build

WORKDIR /app

COPY ./package.json /app

RUN npm install

# 어떤 파일이 이미지에 들어가야 하는지
# 첫 번째 .은 이 프로젝트의 모든 폴더 및 파일들 (Dockerfile을 제외한)
# 두 번째 .은 파일을 저장할 컨테이너 내부 경로 (ex /app)
COPY ./ /app

EXPOSE 6006

RUN npm run build-storybook

CMD ["npm", "run", "storybook"]
```

[Jenkins] Pipeline Script

```
pipeline{
    agent any

    environment {
        Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
    }

    stages{
        stage('GIT CLONE'){
            steps{
                git branch : 'develop-front',
                        credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SETTING ENV'){
            steps{
                dir('dubeng-front'){
                    sh '''
                        cp /home/ubuntu/env/front_server/.env .env
                    '''
                }
            }
        }
        stage('DOCKER BUILD'){
            steps{
                dir('dubeng-front'){
                    sh 'docker build -t dub/storybook -f Dockerfile-storybook .'
                }
            }
        }
        stage('DEPLOY'){
            steps{
                sh '''
                    docker stop storybook-container || true
                    docker rm storybook-container || true
                    docker run --name storybook-container -d -p 6006:6006 dub/storybook
                '''

            }
        }
    }
    // end
    post {
        success {
            mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
        failure {
            mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Detai
            endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
            channel: 'b208-jenkins-notification'
            )
        }
    }
}
```

▼ **Jenkins Mattermost WebHook**

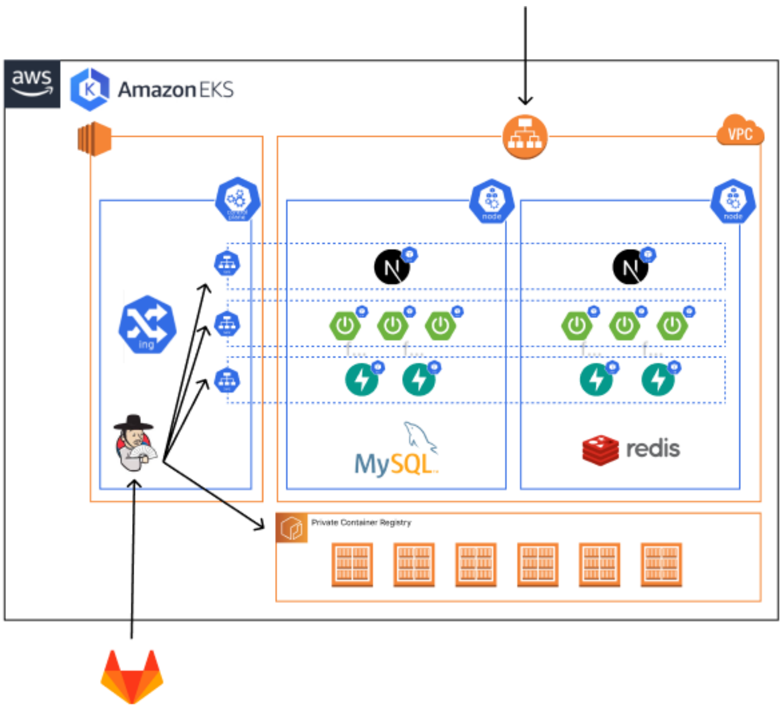
```
environment {
        Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
}
stages{
    ...
}

// end
post {
    success {
        mattermostSend (color: 'good',
        message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)
        endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
        channel: 'b208-jenkins-notification'
        )
    }
    failure {
        mattermostSend (color: 'danger',
        message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)
        endpoint: 'https://meeting.ssafy.com/hooks/ros5qqo1dtykpptm5onqor9gxe',
        channel: 'b208-jenkins-notification'
        )
    }
}
```

[운영 환경] - 쿠버네티스

[도메인] : dub-eng.com

쿠버네티스 아키텍쳐



▼ **AWS ALB ingress.yml**

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    # SSL Settings
    alb.ingress.kubernetes.io/certificate-arn: [arn]
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443},{"HTTP":80}]'
    alb.ingress.kubernetes.io/ssl-redirect: '443'
    alb.ingress.kubernetes.io/enable-cors: "true"
spec:
  rules:
    - host: dub-eng.com
      http:
        paths:
          - pathType: Prefix
            path: /
            backend:
              service:
                name: dubeng-front-service
                port:
                  number: 80
          - pathType: Prefix
            path: /user
            backend:
             service:
               name: dubeng-user-service
               port:
                 number: 80
          - pathType: Prefix
            path: /file
            backend:
              service:
                name: dubeng-filesave-service
                port:
                  number: 80
          - pathType: Prefix
            path: /admin
            backend:
              service:
                name: dubeng-admin-service
                port:
                  number: 80
          - pathType: Prefix
            path: /dub
            backend:
              service:
                name: dubeng-dublist-service
                port:
                  number: 80
          - pathType: Prefix
            path: /recommend
            backend:
              service:
                name: dubeng-recommend-service
                port:
                  number: 80
          - pathType: Prefix
            path: /record
            backend:
              service:
                name: dubeng-dub-service
                port:
                  number: 80
```

▼ **Jenkins Pipeline Sciprt 및 배포 설정 파일**

▼ **Spring - filesave Server**

[Docker] Dockerfile

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar

EXPOSE 9001

COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

[Jenkins] pipeline Script

```
pipeline {
    agent any

    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/filesave',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SPRING BUILD'){
            steps{
                dir('back/storage'){
                    sh '''
                    cp /home/ubuntu/env/filesave-server/application-dev.yml ./src/main/resources/application-dev.yml
                    chmod +x ./gradlew
                    ./gradlew clean build -x test
                    '''
                }
            }
        }
        stage('Image Build'){
            steps{
                dir('back/storage'){
                    sh '''
                        aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin [ac
                        docker rmi -f dub-fileserver:1.0 || true
                        docker rmi -f [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dub-fileserver:1.0 || true
                        docker build -t dub-fileserver:1.0 .
                        docker tag dub-fileserver:1.0 [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dub-fileserver:1.0

                    '''
                }
            }
        }
        stage('ECR PUSH'){
            steps{
                sh '''
                    docker push [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dub-fileserver:1.0
                '''
            }
        }
        stage('KUBECTL APPLY'){
            steps{
                sh '''
                    /var/lib/jenkins/bin/kubectl delete -f /home/ubuntu/kubernetes/filesave-server/filesave.yml || true
                    /var/lib/jenkins/bin/kubectl create -f /home/ubuntu/kubernetes/filesave-server/filesave.yml
                '''
            }
        }
    }
}
```

[kuberenetes] file-service.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-filesave-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: dubeng-filesave-app

  template:
    metadata:
      labels:
        app: dubeng-filesave-app
    spec:
      containers:
        - name: dubeng-filesave-app
          image: [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dub-fileserver:1.0
          imagePullPolicy: Always
          ports:
            - containerPort: 9001
          volumeMounts:
            - mountPath: "/Home"
              name: dubeng-volume
      volumes:
        - name: dubeng-volume
          persistentVolumeClaim:
            claimName: dub-ebs-claim

      imagePullSecrets:
        - name: ecr-secret
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: dubeng-filesave-service
  labels:
    app: dubeng-filesave-app
spec:
  selector:
    app: dubeng-filesave-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9001
```

▼ **Spring - dublist Server**

[Docker] Dockerfile

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar

EXPOSE 9001

COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

[Jenkins] pipeline Script

```
pipeline {
    agent any

    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/dubeng',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SPRING BUILD'){
            steps{
                dir('back/dubeng-dublist'){
                    sh '''
                    cp /home/ubuntu/env/dublist_server/application-dev.yml ./src/main/resources/application-dev.yml
                    chmod +x ./gradlew
                    ./gradlew clean build -x test
                    '''
                }
            }
        }
        stage('Image Build'){
            steps{
                dir('back/dubeng-dublist'){
                    sh '''
                        aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin [ac
                        docker rmi -f dubeng-dublist:1.0 || true
                        docker rmi -f [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-dublist:1.0 || true
                        docker build -t dubeng-dublist:1.0 .
                        docker tag dubeng-dublist:1.0 [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-dublist:1.0

                    '''
                }
            }
        }
        stage('ECR PUSH'){
            steps{
                sh '''
                    docker push [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-dublist:1.0
                '''
            }
        }
        stage('KUBECTL APPLY'){
            steps{
                sh '''
                    /var/lib/jenkins/bin/kubectl delete -f /home/ubuntu/kubernetes/dublist-server/dublist.yml || true
                    /var/lib/jenkins/bin/kubectl create -f /home/ubuntu/kubernetes/dublist-server/dublist.yml
                '''
            }
        }
    }
}
```

[kuberenetes] dublist-service.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-dublist-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: dubeng-dublist-app

  template:
    metadata:
      labels:
        app: dubeng-dublist-app
    spec:
      containers:
        - name: dubeng-dublist-app
          image: [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-dublist:1.0
          imagePullPolicy: Always
          ports:
            - containerPort: 8080

      imagePullSecrets:
        - name: ecr-secret

---
apiVersion: v1
kind: Service
metadata:
  name: dubeng-dublist-service
  labels:
    app: dubeng-dublist-app
spec:
  selector:
    app: dubeng-dublist-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

▼ **Spring - User Server**

[Docker] Dockerfile

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar

EXPOSE 9001

COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

[Jenkins] pipeline Script

```
pipeline {
    agent any

    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/user',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SPRING BUILD'){
            steps{
                dir('back/dubeng-user'){
                    sh '''
                    cp /home/ubuntu/env/user_server/application-dev.yml ./src/main/resources/application-dev.yml
                    chmod +x ./gradlew
                    ./gradlew clean build -x test
                    '''
                }
            }
        }
        stage('Image Build'){
            steps{
                dir('back/dubeng-user'){
                    sh '''
                        aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin [ac
                        docker rmi -f dub-eng-user:1.0 || true
                        docker rmi -f [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dub-eng-user:1.0 || true
                        docker build -t dub-eng-user:1.0 .
                        docker tag dub-eng-user:1.0 [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dub-eng-user:1.0

                    '''
                }
            }
        }
        stage('ECR PUSH'){
            steps{
                sh '''
                    docker push [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dub-eng-user:1.0
                '''
            }
        }
        stage('KUBECTL APPLY'){
            steps{
                sh '''
                    /var/lib/jenkins/bin/kubectl delete -f /home/ubuntu/kubernetes/user-server/ || true
                    /var/lib/jenkins/bin/kubectl create -f /home/ubuntu/kubernetes/user-server/
                '''
            }
        }
    }
}
```

[kuberenetes] user-service.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: dubeng-user-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: dubeng-user-app

  template:
    metadata:
      labels:
        app: dubeng-user-app
    spec:
      containers:
        - name: dubeng-user-app
          image: [accountID].dkr.ecr.ap-northeast-2.amazonaws.com/dub-eng-user:1.0
          imagePullPolicy: Always
          ports:
            - containerPort: 9000
      imagePullSecrets:
        - name: ecr-secret
---

apiVersion: v1
kind: Service
metadata:
  name: dubeng-user-service
  labels:
    app: dubeng-user-app
spec:
  selector:
    app: dubeng-user-app
  ports:
    - protocol : TCP
      port: 80
      targetPort: 9000
```

▼ **conda - Admin**

[Dockerfile]

```
FROM continuumio/miniconda:latest

WORKDIR /app

COPY . .

RUN chmod +x boot.sh

RUN conda env create -f environment.yml

RUN echo "source activate admin-environment" >> ~/.bashrc
ENV PATH /opt/conda/envs/admin-environment/bin:$PATH

RUN apt-get --allow-releaseinfo-change update
RUN apt-get install -y ffmpeg

RUN pip install -r requirements.txt

EXPOSE 5000

ENTRYPOINT ["./boot.sh"]
```

[environment.yml]

```
name: admin-environment
channels:
- defaults
dependencies:
- python=3.8
- flask
- gunicorn
```

boot.sh

실행에 필요한 쉘 스크립트 파일 (gunicorn 설정)

```
#!/bin/sh
# -t 240 : timeout 설정
exec gunicorn -b :5000 --access-logfile - -t 240 --error-logfile - app:app
```

requirements.txt

pipeline 모듈 설치를 위한 config 파일

```
pip freeze > requirements.txt
```

▼ **requirements.txt**

```
absl-py==1.4.0
anyio==3.6.2
asttokens==2.2.1
astunparse==1.6.3
```

```
audioread==3.0.0
backcall==0.2.0
blinker==1.6.2
boto3==1.26.121
botocore==1.29.121
cachetools==5.3.0
certifi==2022.12.7
cffi==1.15.1
charset-normalizer==3.1.0
click==7.1.2
colorama==0.4.6
decorator==5.1.1
executing==1.2.0
ffmpeg-python==0.2.0
Flask==2.0.0
flatbuffers==23.3.3
future==0.18.3
gast==0.4.0
google-api-core==2.11.0
google-api-python-client==2.86.0
google-auth==2.17.3
google-auth-httplib2==0.1.0
google-auth-oauthlib==1.0.0
google-pasta==0.2.0
googleapis-common-protos==1.59.0
grpcio==1.54.0
h11==0.12.0
h2==4.1.0
h5py==3.8.0
hpack==4.0.0
httpcore==0.13.7
httplib2==0.22.0
httpx==0.19.0
hyperframe==6.0.1
idna==3.4
importlib-metadata==6.6.0
ipython==8.12.0
itsdangerous==2.1.2
jax==0.4.8
jedi==0.18.2
Jinja2==3.1.2
jmespath==1.0.1
joblib==1.2.0
keras==2.12.0
libclang==16.0.0
librosa==0.8.1
llvmlite==0.38.1
Markdown==3.4.3
MarkupSafe==2.1.2
matplotlib-inline==0.1.6
ml-dtypes==0.1.0
norbert==0.2.1
numba==0.55.2
numpy==1.22.4
oauthlib==3.2.2
opt-einsum==3.3.0
packaging==23.1
pandas==1.5.3
parso==0.8.3
pickleshare==0.7.5
platformdirs==3.2.0
pooch==1.7.0
prompt-toolkit==3.0.38
protobuf==3.20.3
pure-eval==0.2.2
pyasn1==0.5.0
pyasn1-modules==0.3.0
pycparser==2.21
pydub==0.25.1
Pygments==2.15.1
PyMySQL==1.0.3
pyparsing==3.0.9
python-dateutil==2.8.2
pytube==12.1.3
pytz==2023.3
requests==2.28.2
requests-oauthlib==1.3.1
resampy==0.4.2
rfc3986==1.5.0
rsa==4.9
s3transfer==0.6.0
scikit-learn==1.2.2
scipy==1.10.1
six==1.16.0
sniffio==1.3.0
soundfile==0.12.1
spleeter==2.3.2
```

```
stack-data==0.6.2
termcolor==2.2.0
threadpoolctl==3.1.0
traitlets==5.9.0
typer==0.3.2
typing_extensions==4.5.0
uritemplate==4.1.1
urllib3==1.26.15
waitress==2.1.2
wcwidth==0.2.6
Werkzeug==2.3.1
wrapt==1.14.1
xmltodict==0.13.0
youtube-transcript-api==0.6.0
zipp==3.15.0
```

python env 환경 변수

```
#환경 변수 파일은 아래의 경로에 위치한다.
/home/ubuntu/env/admin_server/env-vedioInfo.txt
/home/ubuntu/env/admin_server/env.txt

# Jenkins 실행 시, 이미지 빌드 전 환경변수 파일을 import 해준다.
cp /home/ubuntu/env/admin_server/env-vedioInfo.txt /var/lib/jenkins/workspace/dub-admin-server/back/dubeng-admin/env-vedioInf
cp /home/ubuntu/env/admin_server/env.txt /var/lib/jenkins/workspace/dub-admin-server/back/dubeng-admin/env.txt
```

Jenkins Script

```
pipeline {
    agent any

    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/admin',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('ENV SETTING'){
            steps{
                dir('back/dubeng-admin'){
                    sh '''
                    cp /home/ubuntu/env/admin_server/env.txt ./env.txt
                    cp /home/ubuntu/env/admin_server/env-vedioInfo.txt ./env-vedioInfo.txt
                    '''
                }
            }
        }
        stage('Image Build'){
            steps{
                dir('back/dubeng-admin'){
                    sh '''
                        aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin [ac
                        docker rmi -f dubeng-admin:1.0 || true
                        docker rmi -f [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-admin:1.0 || true
                        docker build -t dubeng-admin:1.0 .
                        docker tag dubeng-admin:1.0 [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-admin:1.0

                    '''
                }
            }
        }
        stage('ECR PUSH'){
            steps{
                sh '''
                    docker push [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-admin:1.0
                '''
            }
        }
        stage('KUBECTL APPLY'){
            steps{
                sh '''
                    /var/lib/jenkins/bin/kubectl delete -f /home/ubuntu/kubernetes/admin-server/dubeng-admin.yml || true
                    /var/lib/jenkins/bin/kubectl create -f /home/ubuntu/kubernetes/admin-server/dubeng-admin.yml
                '''
            }
        }
    }
}
```

[Kubernetes] admin.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: conda-admin-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dubeng-admin-app

  template:
    metadata:
      labels:
        app: dubeng-admin-app
    spec:
      containers:
        - name: dubeng-admin-app
          image: [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-admin:1.0
          imagePullPolicy: Always
          ports:
            - containerPort: 5000
          volumeMounts:
            - mountPath: "/download/dwn"
              name: dubeng-volume
      volumes:
        - name: dubeng-volume
          persistentVolumeClaim:
            claimName: dub-ebs-claim
      imagePullSecrets:
        - name: ecr-secret
---
apiVersion: v1
kind: Service
metadata:
  name: dubeng-admin-service
  labels:
    app: dubeng-admin-app
spec:
  selector:
    app: dubeng-admin-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5000
```

▼ **FastAPI - dubeng Server**

[Docker] Dockerfile

```
# 베이스가 되는 Docker Image로 python 이미지를 사용
FROM python:3.8-slim

# 처음 실행 시 사용 되는 경로 정보 입니다.
WORKDIR /app

# 현재 경로의 main.py 및 모든 파일을 /app 경로로 복사합니다.
COPY . /app

# 현재 경로의 requirements.txt를 /app 경로로 복사합니다.
COPY requirements.txt /app

# 복사 된 requirements.txt를 사용하여 pip로 패키지를 추가합니다.
RUN pip install -r requirements.txt

# ffmpeg 설치
RUN apt-get --allow-releaseinfo-change update

RUN apt-get install -y ffmpeg
```

```
#  uvicorn을 사용하여 main.py의 app을 실행시킵니다.
CMD uvicorn --host=0.0.0.0 --port 5000 main:app
```

[requirements.txt]

```
anyio==3.6.2
blinker==1.6.2
boto3==1.26.127
botocore==1.29.127
certifi==2022.12.7
cffi==1.15.1
charset-normalizer==3.1.0
click==8.1.3
colorama==0.4.6
cryptography==40.0.2
fastapi==0.95.1
h11==0.14.0
idna==3.4
itsdangerous==2.1.2
Jinja2==3.1.2
jmespath==1.0.1
MarkupSafe==2.1.2
pycparser==2.21
pydantic==1.10.7
pydub==0.25.1
PyMySQL==1.0.3
python-dateutil==2.8.2
requests==2.30.0
s3transfer==0.6.1
six==1.16.0
sniffio==1.3.0
starlette==0.26.1
typing_extensions==4.5.0
urllib3==1.26.15
uvicorn==0.22.0
Werkzeug==2.3.3
```

[Jenkins] pipeline Script

```
pipeline {
    agent any

    stages {
        stage('GIT CLONE') {
            steps{
                git branch : 'develop-back/user',
                credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('ENV SETTING'){
            steps{
                dir('back/dubeng-dub'){
                    sh 'cp /home/ubuntu/env/dub_server/env.txt ./env.txt'
                }
            }
        }
        stage('Image Build'){
            steps{
                dir('back/dubeng-dub'){
                    sh '''
                        aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin [ac
                        docker rmi -f dubeng-dub:1.0 || true
                        docker rmi -f [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-dub:1.0 || true
                        docker build -t dubeng-dub:1.0 .
                        docker tag dubeng-dub:1.0 [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-dub:1.0

                    '''
                }
            }
        }
        stage('ECR PUSH'){
            steps{
                sh '''
                    docker push [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-dub:1.0
                '''
            }
        }
        stage('KUBECTL APPLY'){
            steps{
                sh '''
                    /var/lib/jenkins/bin/kubectl delete -f /home/ubuntu/kubernetes/dub-server/dub.yml || true
                    /var/lib/jenkins/bin/kubectl create -f /home/ubuntu/kubernetes/dub-server/dub.yml
                '''
            }
        }
    }
}
```

[kubernetes] dubeng-dub.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-dub-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: dubeng-dub-app

  template:
    metadata:
      labels:
        app: dubeng-dub-app
    spec:
      containers:
        - name: dubeng-dub-app
          image: [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dubeng-dub:1.0
          imagePullPolicy: Always
          ports:
            - containerPort: 5000
          volumeMounts:
            - mountPath: "/Home"
              name: dubeng-volume
      volumes:
        - name: dubeng-volume
          persistentVolumeClaim:
            claimName: dub-ebs-claim

      imagePullSecrets:
        - name: ecr-secret
---
apiVersion: v1
kind: Service
metadata:
  name: dubeng-dub-service
  labels:
    app: dubeng-dub-app
spec:
  selector:
    app: dubeng-dub-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5000
```

▼ **Front - NextJS**

[Docker] Dockerfile

```
FROM node:16-alpine AS build

WORKDIR /app

COPY ./package.json /app

RUN npm install

# 어떤 파일이 이미지에 들어가야 하는지
# 첫 번째 .은 이 프로젝트의 모든 폴더 및 파일들 (Dockerfile을 제외한)
# 두 번째 .은 파일을 저장할 컨테이너 내부 경로 (ex /app)
COPY ./ /app
```

```
EXPOSE 3000

RUN npm run build

CMD ["npm", "run", "start"]
```

[Jenkins] Pipeline Script

```
pipeline{
    agent any


    stages{
        stage('GIT CLONE'){
            steps{
                git branch : 'develop-front',
                        credentialsId : 'lancelot1672' , url : 'https://lab.ssafy.com/s08-final/S08P31B208'
            }
        }
        stage('SETTING ENV'){
            steps{
                dir('dubeng-front'){
                    sh '''
                        cp /home/ubuntu/env/front_server/.env .env
                    '''
                }
            }
        }
        stage('DOCKER BUILD'){
            steps{
                dir('dubeng-front'){
                    sh '''
                        aws ecr get-login-password --region ap-northeast-2 | docker login --username AWS --password-stdin [ac

                        docker build -t dub-front:1.0 -f Dockerfile-next .
                        docker tag dub-front:1.0 [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dub-front:1.0
                    '''
                }
            }
        }
        stage('DOCKER PUSH'){
            steps{
                sh '''

                docker push [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dub-front:1.0

                '''

            }
        }
        stage('KUBECTL APPLY'){
            steps{
                sh '''
                    /var/lib/jenkins/bin/kubectl delete -f /home/ubuntu/kubernetes/front-server/dub-front.yml || true
                    /var/lib/jenkins/bin/kubectl create -f /home/ubuntu/kubernetes/front-server/dub-front.yml
                '''
            }
        }
    }
}
```

[kubernetes] dub-front.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: next-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: dubeng-front-app
  template:
    metadata:
      labels:
        app: dubeng-front-app
    spec:
      containers:
        - name: dubeng-front-app
          image: [accountId].dkr.ecr.ap-northeast-2.amazonaws.com/dub-front:1.0
          imagePullPolicy: Always
          ports:
            - containerPort: 3000
      imagePullSecrets:
        - name: ecr-secret
---

apiVersion: v1
kind: Service
metadata:
  name: dubeng-front-service
  labels:
    app: dubeng-front-app
spec:
  selector:
    app: dubeng-front-app
  ports:
    - protocol : TCP
      port: 80
      targetPort: 3000
```