포팅 매뉴얼

1. 사용 도구

이슈 관리	Jira
형상 관리	GitLab
CI/CD	Docker, Jenkins, SonarQube
디자인	Figma
협업 툴	MatterMost, Notion

2. 개발 환경

Server	Ubuntu 20.04.6 LTS
JDK	Amazon Corretto 17
Nginx	1.18.0
MySQL	Ver 9.1.0
Redis	7.2.4
IntelliJ	2024.6
Visual Studio Code	1.90.2
Python	3.8
Docker	27.3.1
Docker-compose	v2.20.0

3. 설정파일 및 환경변수 정보

EC2 포트 번호

Backend	8081
Frontend	3000
MySQL	3306

Redis	6379
Nginx	8080/443
FastAPI	8008
MongoDB	27017, 27018, 27019

외부 서비스

- Kakao 로그인 API
- Firebase API

Backend

- **Spring Boot**

  pem 키 파일은 별도 설정이 필요합니다.

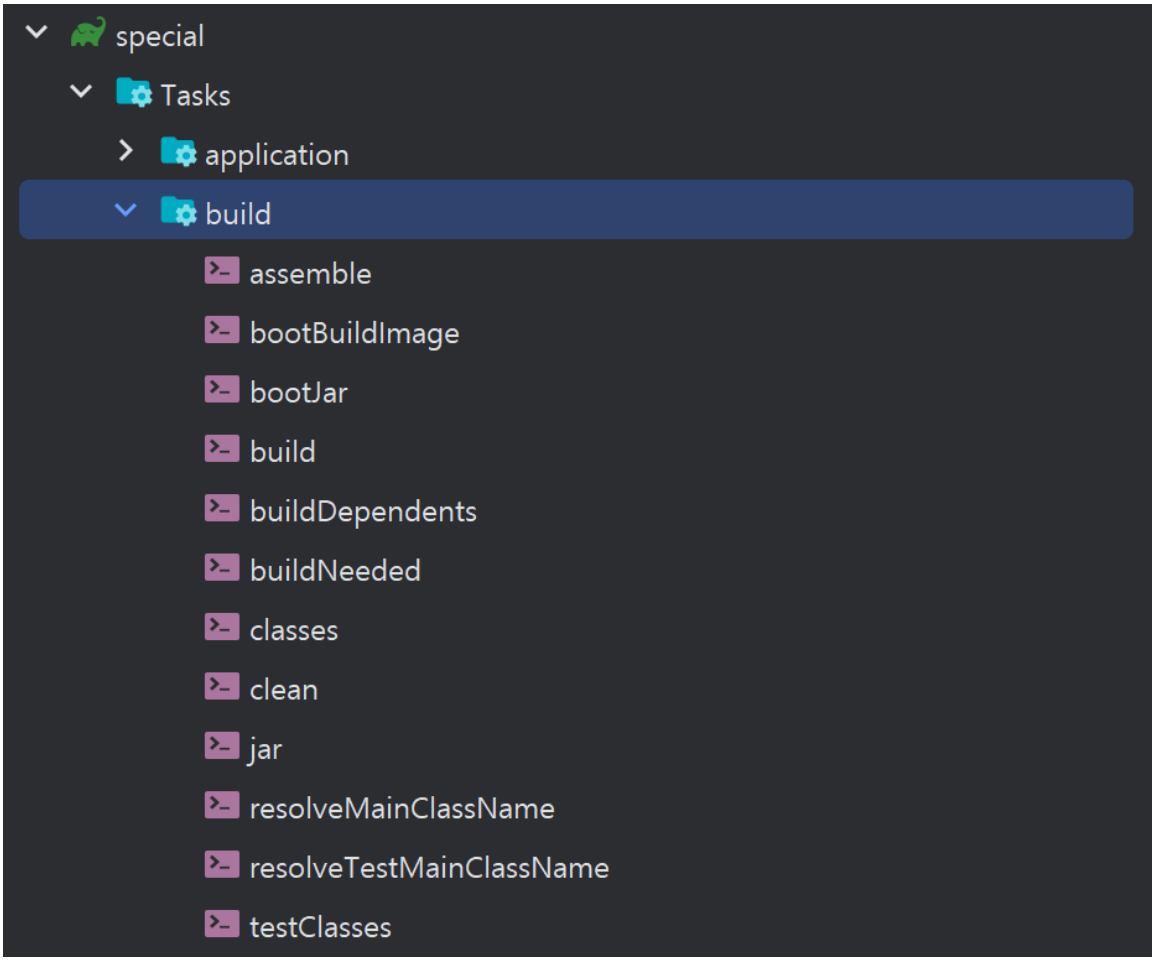
- **FastAPI**

  pem 키 파일은 별도 설정이 필요합니다.

4. 빌드 방법

4-1. Spring Boot

- IntelliJ 우측 코끼리 모양 (Gradle) 클릭
- `clean` → `build`



4-2. FastAPI

```
# root 디렉토리 내에서 실행
# 라이브러리 설치
pip install -r requirements.txt

# 앱 실행 명령어
uvicorn main:app --reload --port 8082
```

4-3. Frontend

```
# 라이브러리 설치
npm install

# android 실행
npm run android
```

```
# android 내에서 app을 실행
cd frontend/android
chmod +x gradlew  # gradlew 파일에 실행 권한 부여

./gradlew assembleRelease  # APK 빌드
```

5. EC2 Setting

▼ 메인 EC2

Docker 설치

- **EC2 서버 접속**

```
ssh -i {KEY_PATH} {USER}@{SERVER_IP}
```

  - KEY_PATH: EC2서버 쪽에서 인증에 사용될 키페어(.pem)파일의 경로
  - USER: 접속한 서버에서 사용할 User계정
    - Ubuntu운영체제를 선택할 경우 기본으로 ubuntu계정 사용
  - SERVER_IP: 접속하고자 하는 서버의 IP주소 *(EC2서버에 부여한 탄력적 IP주소)*

- **패키지 인덱스 업데이트**
  - Ubuntu의 패키지 리스트를 최신 상태로 업데이트

```
sudo apt-get update
```

- **https 관련 패키지 설치 → SSL**
  - `apt`가 HTTPS를 통해 저장소에서 패키지를 다운로드할 수 있도록 필요한 패키지들을 설치

```
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
```

- **Docker의 공식 GPG 키 추가**
  - docker repository 접근을 위한 gpg 키 설정

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

- **docker repository 등록**
  - 시스템의 `apt` 소스 리스트에 Docker의 공식 저장소를 추가

```
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

> 💡 **만일 Docker 저장소 추가는 성공했지만, `certbot` 저장소에서 오류가 발생한다면?**
>
> 1. certbot PPA 제거
>
> ```
> sudo add-apt-repository --remove ppa:certbot/certbot
> sudo apt update
> ```
>
> 2. certbot이 필요하다면 다음 명령어로 설치
>
> ```
> sudo apt update
> sudo apt install certbot
> ```

- **다시 패키지 업데이트**
  - 새로운 저장소를 추가한 후에는 다시 한번 패키지 인덱스를 업데이트

```
sudo apt update
```

- **Docker CE 설치**
  - 이래야 Docker 커뮤니티 에디션 및 CLI 도구 설치 가능

```
sudo apt-get install docker-ce docker-ce-cli containe
rd.io
```

- **Docker 서비스 시작 및 자동 시작 설정**
  - Docker 서비스를 시작하고, 시스템 부팅 시 자동으로 시작되도록 설정

```
sudo systemctl start docker
sudo systemctl enable docker
```

- **Docker 설치 확인**

```
sudo docker --version
sudo docker run hello-world
```

Docker-compose 설치

Jenkins container로 띄울 생각이면, ec2에 설치할 필요 X

Nginx 도커 컨테이너 띄우기

▼ docker에 nginx 이미지 다운로드

```
docker pull nginx:latest
```

▼ docker 컨테이너 실행

```
docker run -d --name nginx -p 80:80 nginx:latest

docker ps # 제대로 설치되었는지 확인
```

Docker 컨테이너 내 Nginx에 SSL/TLS 인증서 적용하기

💡 **Why?**

https 접속을 위해서는 독립된 인증 기관(CA)에서 SSL/TLS 인증서를 획득해야 한다. 웹 사이트가 신뢰를 구축하기 위해 데이터를 교환하기 전에 브라우저와 인증서를 공유하기 때문이다.

도커 컨테이너 내 nginx에 SSL/TLS 인증서를 적용하는 방법

1. 자체 서명된 인증서를 사용하는 방법
2. **신뢰할 수 있는 인증 기관에서 발급된 인증서 사용**
   - `"Let's Encrypt"` **← CA**
   - `Certbot`은 `Let's Encrypt`를 사용해 인증서 생성 / 만료일 갱신

- **Certbot 공식페이지에서는 Snapd를 통한 설치를 권장, 하지만 나는 apt-get**

  💡 **snapd란 ?**

  Ubuntu에서 발표한 일원화된 패키지 관리 툴

- **apt 업데이트**

```
sudo apt-get update
sudo apt-get upgrade
```

- **certbot 설치**

```
sudo apt-get install python3-certbot-nginx
```

- **certbot을 이용하여 도메인에 대한 SSL 인증서 발급**

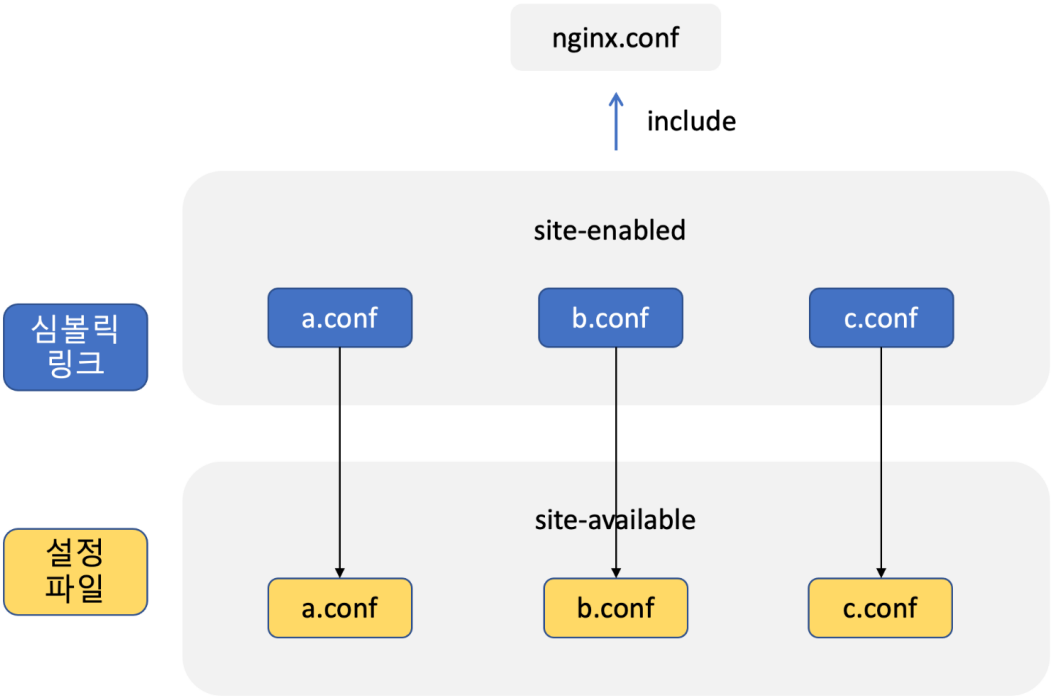
```
sudo certbot certonly --nginx -d ${도메인명}
```

- `/etc/letsencrypt/live/도메인명` **에 제대로 설치되었는지 확인 → ubuntu**

Nginx 설정

- **Nginx 메인 설정파일 설정**

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



- **1. /etc/nginx/sites-available 아래 test.conf 작성**

```
server {
    server_name k11a602.p.ssafy.io;

    location /api/ {
        proxy_pass http://k11a602.p.ssafy.io:8081;
        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;

        # CORS 설정
        add_header 'Access-Control-Allow-Origin' '*'
always;
```

```
        add_header 'Access-Control-Allow-Methods' 'GE
T, POST, OPTIONS, PUT, DELETE';
        add_header 'Access-Control-Allow-Headers' 'Or
igin, X-Requested-With, Content-Type, Accept';

        # OPTIONS 메서드에 대한 CORS Preflight 처리
        if ($request_method = OPTIONS ) {
            add_header 'Access-Control-Allow-Methods'
'GET, POST, OPTIONS, PUT, DELETE';
            add_header 'Access-Control-Allow-Headers'
'Origin, X-Requested-With, Content-Type, Accept';
            add_header 'Access-Control-Max-Age' 17280
00;
            add_header 'Content-Length' 0;
            add_header 'Content-Type' 'text/plain cha
rset=UTF-8';
            return 204;
        }
    }

    location / {
        proxy_pass http://k11a602.p.ssafy.io:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x
_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # CORS 설정 추가
        add_header 'Access-Control-Allow-Origin' '*'
always;
        add_header 'Access-Control-Allow-Credentials'
'true' always;
        add_header 'Access-Control-Allow-Methods' 'GE
T, POST, OPTIONS, PUT, DELETE';
        add_header 'Access-Control-Allow-Headers' 'Or
igin, X-Requested-With, Content-Type, Accept';
```

```
        if ($request_method = OPTIONS ) {
            add_header 'Access-Control-Allow-Origin'
'*';
            add_header 'Access-Control-Allow-Credenti
als' 'true';
            add_header 'Access-Control-Allow-Methods'
'GET, POST, OPTIONS, PUT, DELETE';
            add_header 'Access-Control-Allow-Headers'
'Origin, X-Requested-With, Content-Type, Accept';
            add_header 'Access-Control-Max-Age' 17280
00;
            add_header 'Content-Length' 0;
            add_header 'Content-Type' 'text/plain cha
rset=UTF-8';
            return 204;
        }
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/k11a602.p.s
safy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k11a60
2.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
    if ($host = k11a602.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name k11a602.p.ssafy.io;
    return 404; # managed by Certbot
}
```

- **test.conf 문법 확인**

```
sudo nginx -t
```

- **sites-enabled와 sites-available 사이의 심볼릭 링크**

```
sudo ln -s /etc/nginx/sites-available/test.conf /etc/
nginx/sites-enabled
```

  - sites-enabled에 test.conf 추가된 것 확인

- **Nginx 재구동/재시작**

```
# Nginx 컨테이너 접속 후

nginx -s reload
```

Jenkins 도커 컨테이너

- **Jenkins container 생성 및 구동**

```
cd /home/ubuntu && mkdir jenkins-data

sudo ufw allow 8080/tcp
sudo ufw allow 22/tcp
sudo ufw reload
sudo ufw status

```
sudo docker run -d \
    -p 8080:8080 \
    -v /home/ubuntu/jenkins-data:/var/jenkins_home \
    -v /home/ubuntu/config/.env:/home/ubuntu/config/.
env \
    -v /var/run/docker.sock:/var/run/docker.sock \
    --group-add $(stat -c '%g' /var/run/docker.sock)
\
    --name jenkins \
    jenkins/jenkins
```
```

```

```
sudo docker logs jenkins

sudo docker stop jenkins
sudo docker ps -a
```

- **Jenkins 컨테이너 안에 docker-compose 설치**

```
# Jenkins 컨테이너에 root 권한으로 접근
docker exec -it -u 0 jenkins bash

# Docker CLI도 깔기
apt-get update
apt-get install -y docker.io


# 컨테이너 내부에서 docker-compose 설치
curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# 실행 권한 부여
chmod +x /usr/local/bin/docker-compose
```

- 환경 설정 변경 (중요)

```
cd /home/ubuntu/jenkins-data

mkdir update-center-rootCAs

wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootCA/update-center.crt -O ./update-center-rootCAs/update-center.crt

sudo sed -i 's#https://updates.jenkins.io/update-center.json#https://raw.githubusercontent.com/lework/jenkins-update-center/master/updates/tencent/update-cente
```
```

```
r.json#' ./hudson.model.UpdateCenter.xml

sudo docker restart jenkins
```

- 주요 명령어

```
sudo docker start jenkins
sudo docker stop jenkins
sudo docker logs jenkins
sudo docker logs -f jenkins
```

- config 보안 설정 확인(매우 중요)

```
vi /home/ubuntu/jenkins-data/config.xml

<useSecurity>true</useSecurity>
...(중략)
<securityRealm class="hudson.security.HudsonPrivateSe
curityRealm">
<disableSignup>true</disableSignup>
```

MySQL 컨테이너

- **mysql 데이터 디렉토리 생성**
  - volume 설정을 위함

```
sudo mkdir -p /home/ubuntu/mysql-data

sudo chown 1001:1001 /home/ubuntu/mysql-data  # MySQL
컨테이너 접근 권한 설정
```

MongoDB 컨테이너

- **외부 접속 허용**

  도커 컨테이너 내부에서 파일을 수정할려면 권한이 필요한데 sudo 명령어를 인식하지 못한다.

container 외부에서 mongod.conf 파일을 만들어서 container 내부의 conf 파일을 덮어쓴다.

```
sudo mkdir /home/ubuntu/config/

sudo vim /home/ubuntu/config/mongod.conf
```

```
storage:
  dbPath: /var/lib/mongodb
  journal:
    enabled: true

systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

net:
  port: 27017
  bindIp: 0.0.0.0

processManagement:
  timeZoneInfo: /usr/share/zoneinfo

security:
  authorization: 'enabled'
```

```
docker cp /home/ubuntu/config/mongod.conf mongodb:/et
c/mongod.conf.orig

docker exec -it mongodb bash

cat etc/mongod.conf.orig

sudo ufw allow 27017
```

Redis 컨테이너

- redis 볼륨 설정

```
sudo mkdir /home/ubuntu/redis-data
```

.env파일

- EC2 내 환경변수 파일 등록

```
sudo mkdir /home/ubuntu/config
```

- .env 파일 생성

```
sudo vi /home/ubuntu/config/.env/
```

```
# mySQL
DB_URL=jdbc:mysql://mysql:3306/moa?serverTimezone=UTC
DB_USERNAME=root
DB_PASSWORD=rlarlckd

# kakao
KAKAO_URI=http://k11a602.p.ssafy.io:8080/login/oauth2/code/kakao
KAKAO_CLIENT=9b3cfa9230e5227e66c1bfa9a682323d

# redis
REDIS_HOST=127.0.0.1
REDIS_PORT=6379

# jwt
JWT_SECRET_KEY=9mBtPkj8dLh+3gUdJh2YrmM9w7uNzQ4jY6m9D5u7FH4=

# mongoDB
MONGODB_CONNECTION_STRING: mongodb+srv://jooboy:rlawngud@mycluster.7htkv.mongodb.net/?retryWrites=true&w=majority&appName=myCluster
MONGODB_DATABASE: moa
```

Docker-compose

```
version: '3.8'
services:
  redis:
    image: redis
    container_name: redis
    environment:
      TZ: Asia/Seoul
    networks:
      - mongoCluster
    ports:
      - "6379:6379"  # 호스트와 컨테이너 간의 포트 매핑
    command: ["redis-server", "--bind", "0.0.0.0", "--pr
otected-mode", "no"]  # 모든 IP에서 접근 허용 및 보호 모드 비활
성화
    restart: always  # 컨테이너가 종료되면 자동 재시작

  springboot:
    image: backend
    build: ./backend  # backend 디렉토리에서 Dockerfile을 사
용해 이미지 빌드
    container_name: backend
    env_file: /home/ubuntu/config/.env
    volumes:
      - /home/ubuntu/config/firebase-adminsdk.json:/app/
resources/firebase-adminsdk.json
    environment:
      GOOGLE_APPLICATION_CREDENTIALS: /app/resources/fir
ebase-adminsdk.json
      TZ: Asia/Seoul
    ports:
      - "8081:8081"
    networks:
      - mongoCluster
    depends_on:
      - redis
```

```
      - mysql
      - mongo1
      - mongo2
      - mongo3

  mysql:
    image: mysql:latest
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: rlarlckd
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
    volumes:
      - /home/ubuntu/mysql-data:/var/lib/mysql
    networks:
      - mongoCluster

  mongo1:
    image: mongo:latest
    hostname: mongo1
    container_name: mongo1
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: rlarlckd
    ports:
      - 27017:27017
    volumes:
      - /home/ubuntu/mongo-data1:/data/db
      - /home/ubuntu/key/mongodb.key:/etc/mongodb.key
      - /home/ubuntu/config/mongo-init-replica.sh:/docke
r-entrypoint-initdb.d/mongo-init-replica.sh
    command: 'mongod --replSet myReplicaSet --keyFile /e
tc/mongodb.key --bind_ip_all'
    networks:
      - mongoCluster
```

```
  mongo2:
    image: mongo:latest
    hostname: mongo2
    container_name: mongo2
    restart: always
    depends_on:
      - mongo1
    networks:
      - mongoCluster
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: rlarlckd
    ports:
      - 27018:27017
    volumes:
      - /home/ubuntu/mongo-data2:/data/db
      - /home/ubuntu/key/mongodb.key:/etc/mongodb.key
    command: 'mongod --replSet myReplicaSet --keyFile /e
tc/mongodb.key --bind_ip_all'

  mongo3:
    image: mongo:latest
    hostname: mongo3
    container_name: mongo3
    restart: always
    depends_on:
      - mongo2
    networks:
      - mongoCluster
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: rlarlckd
    ports:
      - 27019:27017
    volumes:
      - /home/ubuntu/mongo-data3:/data/db
      - /home/ubuntu/key/mongodb.key:/etc/mongodb.key
    command: 'mongod --replSet myReplicaSet --keyFile /e
```

```
tc/mongodb.key --bind_ip_all'

networks:
  mongoCluster:
    driver: bridge
```

DockerFile

```
FROM amazoncorretto:17

ENV TZ=Asia/Seoul

WORKDIR /app

COPY build/libs/moa.jar /app/moa.jar

EXPOSE 8081

ENTRYPOINT ["java", "-jar", "/app/moa.jar"]
```

Jenkinsfile

```
pipeline {
    agent any

    stages {

        stage('Checkout SCM') {
            steps {
                echo 'Checking out SCM...'
                checkout scm
            }
        }

        stage('Build JAR') {
            steps {
                echo 'Building JAR file...'
```

```
                sh '''
                    cd backend
                    chmod +x gradlew  # gradlew 파일에 실행
권한 부여
                    ./gradlew clean bootJar --no-build-c
ache -x test
                '''
            }
        }

        stage('Deploy with Docker Compose') {
            steps {
                echo 'Deploying with Docker Compose...'
                sh '''
                    docker-compose down
                    docker-compose up -d --build
                '''
            }
        }
    }

    post {
        always {
            echo 'Pipeline finished.'
        }
        success {
            echo 'Pipeline completed successfully.'
        }
        failure {
            echo 'Pipeline failed.'
        }
    }
}
```

6. dotenv 파일

```
EC2 내 경로: /home/ubuntu/config/.env

# mySQL
DB_URL=jdbc:mysql://<container이름>:3306/<database이름>?se
rverTimezone=UTC
DB_USERNAME=
DB_PASSWORD=

# kakao
KAKAO_URI=
KAKAO_CLIENT=

# Redis
REDIS_HOST=
REDIS_PORT=

# jwt
JWT_SECRET_KEY=

# mongoDB
MONGODB_CONNECTION_STRING=mongodb://<유저ID>:<유저Password
>@k11a602.p.ssafy.io:27017/
MONGODB_DATABASE=

# AWS S3
S3_ACCESS_KEY=
S3_SECRET_KEY=
S3_BUCKET_NAME=

# Firebase
FCM_PROJECT_ID=
GOOGLE_APPLICATION_CREDENTIALS=

# FastAPI
FAST_BASE_URL=
```

▼ GPU EC2

1. AWS EC2 생성

- ubuntu 20.04 LTS / g4dn.xlarge 인스턴스

2. pem키로 EC2 서버 진입

- **EC2 서버 접속**

```
ssh -i {KEY_PATH} {USER}@{SERVER_IP}
```

  - KEY_PATH: EC2서버 쪽에서 인증에 사용될 키페어(.pem)파일의 경로
  - USER: 접속한 서버에서 사용할 User계정
    - Ubuntu운영체제를 선택할 경우 기본으로 ubuntu계정 사용
  - SERVER_IP: 접속하고자 하는 서버의 IP주소 *(EC2서버에 부여한 탄력적 IP주소)*

3. 프로젝트 파일 가져오기

- git 프로젝트 파일 클론

```
git clone <git 레포지토리 url>

# 특정 브랜치 클론할 경우
git clone --branch <브랜치 이름> <git 레포지토리 url>
```

4. 설정 및 패키지 설치

😀 **ubuntu 환경에 설치할 버전**

Python==3.8

CUDA==11.4

cuDNN==9.5.1

1) 기본 패키지 업데이트 및 필수 패키지 설치

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y software-properties-common git wge
t curl build-essential
```

2) Python 3.8 설치

- Python 3.8 설치

```
sudo apt-get install -y python3.8 python3.8-venv pytho
n3.8-dev
```

- update-alternatives로 기본 Python 버전 변경

```
sudo update-alternatives --install /usr/bin/python3 py
thon3 /usr/bin/python3.8 1
sudo update-alternatives --config python3
```

- Python 3.8 버전 확인

```
python3 --version
```

3) CUDA와 cuDNN 설치

(이게 기본.. 다른 방법으로 설치해서 후에 업데이트 할게요...)

- CUDA Toolkit 다운로드 페이지에서 GPU 버전과 Ubuntu 버전에 맞는 설치 파일 다운로드
- 다운로드 및 설치 과정 예시:

  ```
  wget https://developer.download.nvidia.com/compute/
  cuda/11.4.2/local_installers/cuda_11.4.2_470.57.02_
  linux.run
  sudo sh cuda_11.4.2_470.57.02_linux.run
  ```

- 설치 후 환경 변수 설정

```
echo 'export PATH=/usr/local/cuda-11.4/bin:$PATH' >>
~/.bashrc
echo 'export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib6
4:$LD_LIBRARY_PATH' >> ~/.bashrc
source ~/.bashrc
```

- cuDNN도 NVIDIA 개발자 사이트에서 다운로드하고 설치
- **CUDA 버전에 맞는** cuDNN 압축 파일을 다운로드하여 설치 경로 (`/usr/local/cuda/`)에 복사

4) dlib 설치에 필요한 의존성 설치

```
sudo apt install -y cmake libopenblas-dev liblapack-de
v libx11-dev
```

5) CUDA 지원을 위한 dlib 설치

`dlib`을 CUDA 지원으로 설치하려면 `dlib` 소스 코드를 직접 빌드해야 함.

```
git clone https://github.com/davisking/dlib.git
cd dlib
mkdir build
cd build
cmake .. -DDLIB_USE_CUDA=1 -DUSE_AVX_INSTRUCTIONS=1
cmake --build . --config Release
cd ..
python setup.py install --set DLIB_USE_CUDA=1
```

6) 가상환경 설정

- 가상환경 패키지 설치

```
sudo apt install -y python3.8-venv
```

- 가상환경 생성

```
cd <프로젝트 디렉토리>
python3.8 -m venv moa_env
```

- 가상환경 활성화

```
source moa_env/bin/activate
```

- 기본 pip 설치 및 설정

```
sudo apt-get install -y python3-pip
sudo update-alternatives --install /usr/bin/pip pip /u
sr/bin/pip3 1
sudo update-alternatives --config pip
```

- pip 버전 확인

```
pip --version
```

7) Python 패키지 설치

- requirements.txt

```
boto3==1.35.59
cmake==3.30.5
face_recognition==1.3.0
fastapi==0.103.2
imutils==0.5.4
matplotlib==3.5.3
opencv-python==4.10.0.84
Pillow==9.5.0
pydantic==2.9.2
python-dotenv==1.0.1
PyYAML==6.0.1
requests==2.31.0
scikit-image==0.19.3
scipy==1.7.3
torch==1.13.1
```

```
torchvision==0.14.1
tqdm==4.66.6
ultralytics==8.0.145
uvicorn==0.22.0
wheel==0.44.0
```

- requirements.txt로 패키지 설치

```
pip install -r requirements.txt --no-cache-dir
```

5. dotenv 설정

💡 **프로젝트 루트 디렉토리에 위치**

- dotenv

```
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
```

6. 프로젝트 실행

💡 백그라운드 실행을 위해 세션을 만들어서 실행

```
cd <프로젝트 루트 디렉토리>

# screen 세션 만들기
screen -S moa

# 서버 실행 (로그 저장)
uvicorn main:app --host 0.0.0.0 --port 8008 > my_log.log
```

```
2>&1 &

# 세션 나가기 (나가도 서버는 유지됨)
Ctrl+A, D

# 실시간 로그 확인
tail -f my_log.log
```

- 그 외 기타 명령어

```
# 실행 중인 세션 확인
screen -ls

# 세션에 다시 접속
screen -r myserver

# 특정 세션 종료
screen -X -S <세션ID> quit

# 현재 실행중인 프로세스
ps aux | grep uvicorn
```

6. 시연 시나리오

1. 내 얼굴 등록 - 마이페이지

🎤

- 프로필 사진과 닉네임 변경이 가능합니다.
- 여기서 등록하는 얼굴이 추후 AI 사진 분류시 비교 기준으로 사용됩니다.

2. 순간 생성

🎤

- 설명 내용
  - 일회성 모임의 사진을 공유 할 수 있는 곳입니다.
  - 업로드 권한을 설정할 수 있습니다.
    - 나만 올릴 수 있는지
    - 구성원 모두가 올릴 수 있는지

3. 순간 조회

🎤

- 그룹과 달리 순간은 24시간이 지나면 사라집니다.
  - 만료 시간 타이머를 확인할 수 있습니다.
  - 모래시계 애니메이션

4. 그룹 생성

🎤

- 설명 내용
  - 정기적인 모임의 사진을 공유 할 수 있는 곳입니다.
  - 원하는 색상과 이모티콘으로 커스텀할 수 있습니다.
  - 순간과 달리 구성원 모두가 올릴 수 있습니다.

5. 그룹 조회

🎤

- 커스텀 색에 따라 테마가 바뀜
- 그룹에 업로드 된 사진은 24시간의 유효시간을 가집니다.
  - 만료 시간이 임박한 순서대로 사진이 표시됩니다.
  - 만료시간이 표시됩니다.
- 헤더에 vertical 아이콘 클릭
  - PIN번호 확인
  - 관리자면 그룹 삭제 및 수정
  - 멤버면 그룹 나가기
- 카카오톡 초대 링크 발송
  - 그룹 / 순간 입장 → 핀번호 입력
- 사진 업로드
- 사진 분류
  - 내 사진만
    - 마이페이지에서 내가 등록한 얼굴 사진을 기준으로 분류
  - 음식 사진
  - 풍경 사진
- 사진 다운
  - 다운로드할 사진을 선택하고, 다운로드 하면 갤러리에 사진이 저장됩니다.
- 그룹에 사진을 업로드 하면 그룹에 포함된 구성원들에게 업로드 알림이 갑니다.
  - 알림은 포그라운드, 백그라운드 모두 확인할 수 있으며, 백그라운드에서 알림을 클릭시 앱으로 이동합니다.
  - 또한 받은 알림들은 알림 내역에서 최신순으로 확인할 수 있습니다.
  - 정기적인 모임의 사진을 공유 할 수 있는 곳입니다.
  - 원하는 색상과 이모티콘으로 커스텀할 수 있습니다.
  - 순간과 달리 구성원 모두가 올릴 수 있습니다.