포팅 매뉴얼

개발 환경

기본 정보

Backend

NAME	VERSION
JVM	17
Spring Boot	3.2.5
Gradle	8.7
MySQL	8.3.0
Kafka	3.7.0
RabbitMQ	3.13.1

AI

NAME	VERSION
Python	3.10.6
FastAPI	0.111.0
ONNX	-

Frontend

NAME	VERSION
Kotlin	1.9.0
JVM	17
Vue.js	^3.2.13

Embedded

NAME	VERSION
ROS	2 Foxy Fitzroy [Ubuntu 20.04 LTSm window10 지원]
Qt	5.12.8
QtCreator	4.11.0
QMake	3.1
CMake	3.16.3

IDE

NAME	VERSION
Visual Studio Code	1.89.1
Android Studio	2023.2.1
Intellij	2024.1.1 Ultimate

배포 환경

서버 구성

- AWS Lightsail 기준 80 USD 옵션인 16GB 메모리 / vCPU 4개 / 320GS SSD 디스크 / 6TB 전송 인스턴스 또는 이와 대응하는 EC2 인스턴스
    - Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1055-aws x86_64)

서버 환경 설정

방화벽 설정

```
# ufw 설치명령
sudo apt-get install ufw
```

```
# ufw 상태확인 명령
sudo ufw status verbose
sudo ufw status

sudo ufw allow 22 # ssh
sudo ufw enable

# ...
sudo ufw status
```

사용 포트 리스트

PORT	INFO
22	ssh
80	http & Nginx
443	https & Nginx
8080	Backend Server Blue (Spring Boot)
8082	Backend Server Green (Spring Boot)
8000	AI Server Blue (FastAPI)
8001	AI Server Green (FastAPI)
3306	MySQL
10000	Kafka
10001	Kafka
10002	Kafka
8085	Kafka Management Web
5672	RabbitMQ
1883	RabbitMQ & MQTT (TCP)
15672	RabbitMQ Management Web
3000	Frontend Server Blue (Vue.js, WEB)
3001	Frontend Server Green (Vue.js, WEB)
8081	Jenkins

기본 세팅

- Ubuntu의 서버 시간을 UTC에서 KST로 변경
  - AWS의 Ubuntu는 서버 시간 기본값 UTC

```
sudo timedatectl set-timezone Asia/Seoul
```

- 미러 서버를 카카오 서버로 변경
  - 기본 서버가 *.ubuntu.com 이라는 해외망을 이용하는 서버.
  - 국내망을 이용할 수 있는 카카오 미러 서버를 사용.
  - AWS EC2 혹은 AWS Lightsail에서 아래 코드 사용 가능.
    - 타 Ubuntu 서버를 사용할 경우 ap-northeast-2.ec2.archive.ubuntu.com 부분을 "sudo vi /etc/apt/sources.list"으로 확인해서 다른 서버로 변경 후 사용해야 함.

```
sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirror.kakao.com/g' /etc/apt/sources.list
```

- 패키지 목록 업데이트 & 패키지 업데이트
  - 미러 서버가 변경됨. 따라서 update & upgrade 진행.

```
sudo apt-get -y update && sudo apt-get -y upgrade
```

- Swap 영역 할당
  - 용량 확인

    ```
    free -h
    ```

  - 할당 (ex. 4GB)

    ```
    sudo fallocate -l 4G /swapfile
    ```

  - swapfile 권한 수정

    ```
    sudo chmod 600 /swapfile
    ```

  - swapfile 생성

```
sudo mkswap /swapfile
```

- swapfile 활성화

```
sudo swapon /swapfile
```

- 시스템 재부팅 시에도 swap 유지 설정

```
sudo echo '/swapfile none swap sw 0 0' | sudo tee -a
/etc/fstab
```

- 할당 확인

```
free -h
```

Docker 환경 설정

Docker 설치

- 버전 정보
  - Docker
    - 26.0.2
  - Docker Compose
    - 2.26.1
- 필요 Package 설치

```
sudo apt-get -y install apt-transport-https ca-certifica
tes curl gnupg-agent software-properties-common
```

  - apt-transport-https : 패키지 관리자가 https를 통해 데이터 및 패키지에 접근 가능하게 함.
  - ca-certificates : certificate authority에서 발행되는 디지털 서명. SSL 인증서의 PEM 파일이 포함되어 있어 SSL 기반 앱이 SSL 연결이 되어있는지 확인할 수 있다
  - curl : 특정 웹사이트에서 데이터를 다운로드 받을 경우 사용하는 패키지

    - gnupg-agent : OpenPGP 표준 규격의 데이터 통신을 암호화하고 서명할 수 있는 패키지
    - software-properties-common : PPA를 추가하거나 제거할 때 사용
        - PPA : Personal Package Archive(개인 패키지 저장소)를 의미. 캐노니컬社의 우분투에서 기본적으로 제공하는 패키지 외의 사적으로 만든 패키지를 의미
- GPG Key 인증 진행

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

- Repository 등록
    - AMD64

```
sudo add-apt-repository "deb [arch=amd64] https://dow
nload.docker.com/linux/ubuntu $(lsb_release -cs) stab
le"
```

    - ARM64

```
sudo add-apt-repository "deb [arch=arm64] https://dow
nload.docker.com/linux/ubuntu $(lsb_release -cs) stab
le"
```

- Package List 갱신

```
sudo apt-get -y update
```

- Docker Package 설치

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

    - apt-get을 이용하여 설치
        - docker-ce : Docker Community Edition
        - docker-ce-cli : Docker Community Edition의 cli 환경에서 추가로 설치해야 하는 패키지

        - containerd.io : Docker 컨테이너 런타임
- 일반 유저에게 권한 부여
    - Docker는 항상 root로 실행되기 때문에 sudo를 사용하여 명령어를 입력해야 함.
        - 사용자를 docker 그룹에 추가하여 sudo를 사용하지 않아도 docker 명령어 사용 가능.

        ```
        sudo usermod -aG docker ubuntu
        ```

    - Docker 서비스 재시작. 추가적으로 사용자 세션 재로그인 필요

        ```
        sudo service docker restart
        ```

Docker Compose 설치 (Docker 설치 후 Docker Compose가 설치 안 된 경우)

- 버전 정보
    - 2.26.1
- 설치

    ```
    sudo curl -L "https://github.com/docker/compose/releases/download/v2.26.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
    ```

- 권한 변경

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

MySQL

- 버전 정보
    - 8.3.0
- 도커 이미지 받기

    ```
    sudo docker pull mysql:latest
    ```

```
sudo docker pull mysql:8.3.0
```

- 도커 컨테이너 실행

```
docker run -d -p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=비밀번호 \
-e TZ=Asia/Seoul \
-v /var/lib/mysql:/var/lib/mysql \
--name mysql mysql:latest
```

- OS 재시작 후 자동 실행 설정
  - docker-mysql.service 등록

```
sudo vim /etc/systemd/system/docker-mysql.service
```

```
[Unit]
Description=docker-mysql
Wants=docker.service
After=docker.service

[Service]
RemainAfterExit=yes
ExecStart=/usr/bin/docker start mysql
ExecStop=/usr/bin/docker stop mysql

[Install]
WantedBy=multi-user.target
```

  - docker service 활성화

```
sudo systemctl enable docker
```

  - docker service 시작

```
sudo systemctl start docker
```

    - docker-mysql service 활성화

```
sudo systemctl enable docker-mysql.service
```

    - docker-mysql service 시작

```
sudo systemctl start docker-mysql.service
```

- DATABASE(OR SCHEMA) 생성
    - DBMS 사용 / 도커 컨테이너 내부에 접속(`docker exec -it mysql /bin/bash`)하여 `mysql -u root -p` 사용

```
CREATE DATABASE 데이터베이스명 DEFAULT CHARACTER SET utf8mb4;
```

- 계정 생성 및 권한 부여
    - DBMS 사용 / 도커 컨테이너 내부에 접속(`docker exec -it mysql /bin/bash`)하여 `mysql -u root -p` 사용
    - 사용자 계정 생성

```
CREATE USER '계정명'@'%' IDENTIFIED BY '비밀번호';
```

    - 사용자 계정 삭제

```
DROP USER 계정이름@아이피주소;
```

    - 사용자에 데이터베이스 사용 권한 부여

```
GRANT ALL PRIVILEGES ON 데이터베이스명.* TO '계정명'@'%';
```

    - 변경 내용 반영

```
FLUSH PRIVILEGES;
```

- DDL

```
create table if not exists code_group
(
    group_name        varchar(32)  not null
        primary key,
    group_description varchar(128) not null
)
    collate = utf8mb3_unicode_ci;

create table if not exists code_item
(
    id               int auto_increment
        primary key,
    group_name       varchar(255) not null,
    item_name        varchar(255) not null,
    item_description varchar(255) not null,
    constraint code_item_ibfk_1
        foreign key (group_name) references code_group
(group_name)
            on update cascade on delete cascade
)
    collate = utf8mb3_unicode_ci;

create index group_name
    on code_item (group_name);

create table if not exists store
(
    id         int auto_increment
        primary key,
    store_name varchar(255) null,
    hour_fee   int          not null,
    latitude   varchar(20)  null,
    longitude  varchar(20)  null,
    address    varchar(30)  null,
    hours      varchar(20)  null,
    phone      varchar(13)  null,
    created_at datetime(6)  null,
    updated_at datetime(6)  null,
```

```
    deleted_at datetime(6)  null
)
    collate = utf8mb3_unicode_ci;

create table if not exists menu
(
    id           int auto_increment
        primary key,
    store_id     int          null,
    menu_name    varchar(255) null,
    menu_type    char         not null,
    menu_price   int          not null,
    is_available int          not null,
    menu_img_url varchar(255) null,
    created_at   datetime(6)  null,
    updated_at   datetime(6)  null,
    deleted_at   datetime(6)  null,
    constraint FK4sgenfcmk1jajhgctnkpn5erg
        foreign key (store_id) references store (id)
)
    collate = utf8mb3_unicode_ci;

create table if not exists room
(
    id          int auto_increment
        primary key,
    store_id    int          not null,
    coordinatex varchar(255) not null,
    coordinatey varchar(255) not null,
    room_number int          not null,
    fcm_token   varchar(255) null,
    iot_id      varchar(255) null,
    created_at  datetime(6)  null,
    updated_at  datetime(6)  null,
    deleted_at  datetime(6)  null,
    constraint FKpviy6dvvot2ne7jl43j603mn3
        foreign key (store_id) references store (id)
)
```

```
    collate = utf8mb3_unicode_ci;

create table if not exists customer
(
    id         int auto_increment
        primary key,
    room_id    int           null,
    is_theme   int           not null,
    people_num int           not null,
    start_time datetime(6)   null,
    end_time   datetime(6)   null,
    room_fee   int default 0 not null,
    constraint FK9q5moa3m32falsvehbvmb8ons
        foreign key (room_id) references room (id)
)
    collate = utf8mb3_unicode_ci;

create table if not exists order_menu
(
    id           bigint auto_increment
        primary key,
    customer_id  int         null,
    order_status int         not null,
    created_at   datetime(6) null,
    delivered_at datetime(6) null,
    constraint FKi7hcklrx4d9reh9nw8hlxdj79
        foreign key (customer_id) references customer (i
d)
)
    collate = utf8mb3_unicode_ci;

create table if not exists order_menu_detail
(
    id            bigint auto_increment
        primary key,
    menu_id       int    null,
    order_menu_id bigint null,
    quantity      int    not null,
```

```
    unit_price    int    not null,
    total_price   int    not null,
    constraint FKfrjqv1saig4cqalrf50qj0pi7
        foreign key (menu_id) references menu (id),
    constraint FKh58x09597sk2myb8dd71bh15o
        foreign key (order_menu_id) references order_men
u (id)
)
    collate = utf8mb3_unicode_ci;

create table if not exists order_menu_status_log
(
    id            bigint auto_increment
        primary key,
    order_menu_id bigint      null,
    before_status int         not null,
    after_status  int         not null,
    changed_at    datetime(6) null,
    constraint FKatmw07yjm8p1duu4wuxeermc1
        foreign key (order_menu_id) references order_men
u (id)
)
    collate = utf8mb3_unicode_ci;

create table if not exists theme
(
    id              int auto_increment
        primary key,
    theme_type      varchar(255) null,
    theme_video_url varchar(255) null,
    created_at      datetime(6)  null,
    updated_at      datetime(6)  null,
    deleted_at      datetime(6)  null
)
    collate = utf8mb3_unicode_ci;

create table if not exists game
(
```

```
    id              int auto_increment
        primary key,
    game_name       varchar(255) not null,
    game_detail     varchar(255) not null,
    min_player      int          not null,
    max_player      int          not null,
    min_playtime    int          not null,
    max_playtime    int          not null,
    game_difficulty float        not null,
    game_img_url    varchar(255) null,
    theme_id        int          null,
    created_at      datetime(6)  null,
    updated_at      datetime(6)  null,
    deleted_at      datetime(6)  null,
    constraint FKox6tdvw0mm5klagbu2jgq1w6j
        foreign key (theme_id) references theme (id)
)
    collate = utf8mb3_unicode_ci;

create table if not exists game_tag_category
(
    id           int auto_increment
        primary key,
    game_id      int          null,
    group_name   varchar(255) null,
    code_item_id int          null,
    constraint FKt21xw6v8larc7f2fnrm5kou9u
        foreign key (game_id) references game (id),
    constraint game_tag_category_ibfk_1
        foreign key (group_name) references code_group
(group_name),
    constraint game_tag_category_ibfk_2
        foreign key (code_item_id) references code_item
(id)
)
    collate = utf8mb3_unicode_ci;

create index code_item_id
```

```
    on game_tag_category (code_item_id);

create index group_name
    on game_tag_category (group_name);

create table if not exists stock
(
    id             int auto_increment
        primary key,
    store_id       int          null,
    game_id        int          null,
    is_available   int          not null,
    stock_location varchar(255) null,
    created_at     datetime(6)  null,
    updated_at     datetime(6)  null,
    deleted_at     datetime(6)  null,
    constraint FKa9un5jymd7dqr3orbkt9fa74b
        foreign key (store_id) references store (id),
    constraint FKmjukt6ibaqa0vo6dotmw4md2w
        foreign key (game_id) references game (id)
)
    collate = utf8mb3_unicode_ci;

create table if not exists order_game
(
    id           bigint auto_increment
        primary key,
    customer_id  int         null,
    stock_id     int         null,
    order_type   int         not null,
    order_status int         not null,
    created_at   datetime(6) null,
    delivered_at datetime(6) null,
    constraint FK73tohaymnx4vgrfnpiswbafks
        foreign key (stock_id) references stock (id),
    constraint FKsckwl4e01euj7n5va07d9csro
        foreign key (customer_id) references customer (i
d)
```

```
)
    collate = utf8mb3_unicode_ci;

create table if not exists order_game_status_log
(
    id            bigint auto_increment
        primary key,
    order_game_id bigint      null,
    before_status int         not null,
    after_status  int         not null,
    changed_at    datetime(6) null,
    constraint FKps2iblfbq17edi3oxh1u2c55w
        foreign key (order_game_id) references order_gam
e (id)
)
    collate = utf8mb3_unicode_ci;

create table if not exists turtle
(
    id         int auto_increment
        primary key,
    store_id   int         null,
    is_working int         not null,
    created_at datetime(6) null,
    updated_at datetime(6) null,
    deleted_at datetime(6) null,
    constraint FK26x2bj6hp63i6ubq4a8blylku
        foreign key (store_id) references store (id)
)
    collate = utf8mb3_unicode_ci;

create table if not exists turtle_log
(
    id                 bigint auto_increment
        primary key,
    turtle_id          int          null,
    command_type       int          not null,
    command_start_time datetime(6)  null,
```

```
    command_end_time   datetime(6)  null,
    is_success         int          not null,
    log_message        varchar(255) null,
    order_menu_id      bigint       null,
    order_game_id      bigint       null,
    constraint FK5l95b97xpshjjtb2caoco4i4i
        foreign key (order_menu_id) references order_men
u (id),
    constraint FKcuba1vw0rit7bfddfbeggdnyd
        foreign key (turtle_id) references turtle (id),
    constraint FKoi2f50g2qn6m56i2ldl6u3ebh
        foreign key (order_game_id) references order_gam
e (id)
)
    collate = utf8mb3_unicode_ci;
```

Nginx & Certbot SSL 인증

- 버전 정보
  - Nginx
    - 1.25.5
  - Certbot
    - 2.10.0
- 도커 이미지 받기
  - Nginx

    ```
    sudo docker pull nginx:latest
    ```

    ```
    sudo docker pull nginx:1.25.5
    ```

  - Certbot

    ```
    sudo docker pull certbot/certbot:latest
    ```

```
sudo docker pull certbot/certbot:v2.10.0
```

- 도커 컨테이너 실행 및 SSL 인증서 발급 & HTTPS 적용

```
vi docker-compose.yml
```

```
# docker compose v2.25.0 이상에서는 version 명시 안 해도 됨
# version: '3.9'
services:
  nginx:
    image: nginx:latest
    volumes:
      - ./conf/nginx.conf:/etc/nginx/nginx.conf
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    restart: always
    ports:
      - 80:80
      - 443:443
    container_name: nginx
  certbot:
    image: certbot/certbot:latest
    restart: unless-stopped
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    container_name: certbot
```

`docker-compose.yml`이 있는 디렉토리에서 `conf` 디렉토리 생성 후 `nginx.conf` 파일 작성한다.

```
mkdir conf
vi conf/nginx.conf
```

```
server {
     listen 80;
```

```
    server_name [구매한 도메인];

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

```
docker-compose -f docker-compose.yml up -d
```

docker compose로 컨테이너 실행 후 정상적으로 동작된다면 아래에 진행 과정을 진행한다.

```
vi init-letsencrypt.sh
```

아래 코드에서 괄호부분에 도메인, 아까 Certbot 경로, 이메일을 넣어준다.

```
#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
  echo 'Error: docker-compose is not installed.' >&2
  exit 1
fi

domains=([구매한 도메인] www.[구매한 도메인])
rsa_key_size=4096
data_path="./[CUSTOM]/certbot"
email="[이메일]" # Adding a valid address is strongly rec
ommended
staging=0 # Set to 1 if you're testing your setup to avo
id hitting request limits

if [ -d "$data_path" ]; then
  read -p "Existing data found for $domains. Continue an
d replace existing certificate? (y/N) " decision
  if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; t
hen
```

```
    exit
  fi
fi


if [ ! -e "$data_path/conf/options-ssl-nginx.conf" ] ||
[ ! -e "$data_path/conf/ssl-dhparams.pem" ]; then
  echo "### Downloading recommended TLS parameters ..."
  mkdir -p "$data_path/conf"
  curl -s https://raw.githubusercontent.com/certbot/cert
bot/master/certbot-nginx/certbot_nginx/_internal/tls_con
figs/options-ssl-nginx.conf > "$data_path/conf/options-s
sl-nginx.conf"
  curl -s https://raw.githubusercontent.com/certbot/cert
bot/master/certbot/certbot/ssl-dhparams.pem > "$data_pat
h/conf/ssl-dhparams.pem"
  echo
fi

echo "### Creating dummy certificate for $domains ..."
path="/etc/letsencrypt/live/$domains"
mkdir -p "$data_path/conf/live/$domains"
docker-compose run --rm --entrypoint "\
  openssl req -x509 -nodes -newkey rsa:$rsa_key_size -da
ys 1\
    -keyout '$path/privkey.pem' \
    -out '$path/fullchain.pem' \
    -subj '/CN=localhost'" certbot
echo


echo "### Starting nginx ..."
docker-compose up --force-recreate -d nginx
echo

echo "### Deleting dummy certificate for $domains ..."
docker-compose run --rm --entrypoint "\
  rm -Rf /etc/letsencrypt/live/$domains && \
```

```
  rm -Rf /etc/letsencrypt/archive/$domains && \
  rm -Rf /etc/letsencrypt/renewal/$domains.conf" certbot
echo


echo "### Requesting Let's Encrypt certificate for $doma
ins ..."
#Join $domains to -d args
domain_args=""
for domain in "${domains[@]}"; do
  domain_args="$domain_args -d $domain"
done

# Select appropriate email arg
case "$email" in
  "") email_arg="--register-unsafely-without-email" ;;
  *) email_arg="--email $email" ;;
esac

# Enable staging mode if needed
if [ $staging != "0" ]; then staging_arg="--staging"; fi

docker-compose run --rm --entrypoint "\
  certbot certonly --webroot -w /var/www/certbot \
    $staging_arg \
    $email_arg \
    $domain_args \
    --rsa-key-size $rsa_key_size \
    --agree-tos \
    --force-renewal" certbot
echo

echo "### Reloading nginx ..."
docker-compose exec nginx nginx -s reload
```

```
sudo ./init-letsencrypt.sh
```

정상적으로 실행이 된다면 SSL 인증서 발급이 완료된 것이다. 그러면 HTTPS 적용으로 넘어가자.

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

```
server {
        listen 80;
        server_name [도메인 이름];
        server_tokens off;

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


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

}
server {
        listen 443 ssl;
        server_name [도메인 이름];
        server_tokens off;

        ssl_certificate /etc/letsencrypt/live/[도메인 이
름]/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/[도메인
이름]/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        location / {
        ...
}
```

```
vi docker-compose.yml
```

```
# docker compose v2.25.0 이상에서는 version 명시 안 해도 됨
# version: '3.9'
services:
  nginx:
    image: nginx:latest
    volumes:
      - ./conf/nginx.conf:/etc/nginx/nginx.conf
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
      - ./data/nginx/conf.d:/etc/nginx/conf.d
    restart: always
    ports:
      - 80:80
      - 443:443
    container_name: nginx
    command: '/bin/sh -c ''while :; do sleep 6h & wait
$${!}; nginx -s reload; done & nginx -g "daemon off;"'''
  certbot:
    image: certbot/certbot:latest
    restart: unless-stopped
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    container_name: certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do
certbot renew; sleep 12h & wait $${!}; done;'"
```

```
docker compose -f docker-compose.yml up -d
```

정상적으로 실행되면 HTTPS 적용이 될 것이다.

- 추가 설정 (리버스 프록시 등)

```
/etc/nginx/conf.d/be-service-url.inc
```

```
set $be_service_url http://도메인:8080;
```

```
/etc/nginx/conf.d/fe-service-url.inc
```

```
set $fe_service_url http://도메인:3000;
```

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

```
server {
    server_name 도메인;

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

  include /etc/nginx/conf.d/be-service-url.inc;
  include /etc/nginx/conf.d/fe-service-url.inc;
  include /etc/nginx/conf.d/ai-service-url.inc;

  # file upload
  client_max_body_size 100M;

    location /api {
        proxy_pass $be_service_url;
        proxy_redirect off;
        charset utf-8;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_fo
rwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-NginX-Proxy "true";
    }

    location /kafka-messages {
        proxy_pass $be_service_url;
```

```
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwar
ded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_read_timeout 86400; # 24 hour
    }

    location /ai {
        proxy_pass $ai_service_url;
        proxy_redirect off;
        charset utf-8;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_fo
rwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-NginX-Proxy "true";
    }

    location / {
        proxy_pass $fe_service_url;
        proxy_redirect off;
        charset utf-8;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_fo
rwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-NginX-Proxy "true";
    }
```

```
    #error_page 404      /404. html;
    # redirect server error pages to the static page /50
x. html
    #
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.
0.0.1:80
    #
    #location ~ \. php$ {
    #   proxy_pass http://127.0.0.1;
    #}

    # pass the PHP scripts to FastGI server listening on
127.0.0.1:9000
    #
    #location ~ \. php$ {
    #   root html;
    #   fastcgi_pass 127.0.0.1:9000;
    #   fastcgi_index index.php;
    #   fastegi_param SCRIPT_FILENAME /scripts$fastcgi_s
cript_name;
    #   include fastegi_params;
    #}

    # deny access to htaccess files, if Apache's documen
t root
    # concurs with nginx's one
    #
    #location ~ A.ht {
    #   deny all;
    #}

    listen [::]:443 ssl ipv6only=on; # managed by Certbo
t
```

```
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/도메인/fullchai
n.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/도메인/priv
key.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # m
anaged by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # man
aged by Certbot

}

server {
    if ($host = 도메인) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;
    server_name 도메인;
    return 404; # managed by Certbot
}
```

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

```
events {}
http {
        resolver 8.8.8.8 valid=10s;

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

        server {
        listen 80;
        server_name 도메인;
        server_tokens off;

        location /.well-known/acme-challenge/ {
```

```
                root /var/www/certbot;
        }
        }
        server {
        listen 443 ssl;
        server_name 도메인;
        server_tokens off;

        ssl_certificate /etc/letsencrypt/live/도메인/full
chain.pem;
        ssl_certificate_key /etc/letsencrypt/live/도메인/
privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        location / {

        }
        }
}
```

- 설정이 변경되면 nginx를 재시작 또는 reload 필요.

Jenkins

- 버전 정보
  - 2.440.3
- 도커 이미지 받기

```
sudo docker pull jenkins/jenkins:lts-jdk17
```

```
sudo docker pull jenkins/jenkins:2.440.3-jdk17
```

- 도커 컨테이너 실행

```
vi docker-compose-jenkins.yml
```

```
# docker compose v2.25.0 이상에서는 version 명시 안 해도 됨
# version: '3.9'
services:
  jenkins:
    image: jenkins/jenkins:lts-jdk17
    user: root
    ports:
      - "8081:8080"
    environment:
      - JENKINS_OPTS=--httpPort=8080
      - TZ=Asia/Seoul
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/local/bin/docker-compose:/usr/local/bin/doc
ker-compose
    restart: unless-stopped
    container_name: jenkins
```

```
docker compose -f docker-compose-jenkins.yml up -d
```

- 초기 설정
  - http://도메인:8081 로 접속
  - 암호 입력

    ```
    # 방법 1 : log를 통해 암호 확인.
    docker logs jenkins
    # 방법 2 : 컨테이너 내부에 접속. 암호 파일을 확인.
    docker exec -it jenkins /bin/bash
    sudo cat /var/lib/jenkins/secrets/initialAdminPasswor
    d
    ```

  - 계정 생성
- 추가 설정

- Credential 설정
  - Git Token 등과 같은 Credential을 등록한다.
- Plugin 설치

```
# ssh 커맨드 입력에 사용
SSH Agent

# docker 이미지 생성에 사용
Docker
Docker Commons
Docker Pipeline
Docker API

# 웹훅을 통해 브랜치 merge request 이벤트 발생시 Jenkins 자동 빌드에 사용
Generic Webhook Trigger

# 타사 레포지토리 이용시 사용 (GitLab, Github 등)
GitLab
GitLab API
GitLab Authentication
GitHub Authentication

# Node.js 빌드시 사용
NodeJS
```

  - 위의 Plugin들을 비롯하여 필요한 Plugin을 설치한다.

Kafka

- 버전 정보
  - kafka
    - 3.7.0
    - apache/kafka, bitnami/kafka 등 3.7.0 기반이면 어느 이미지를 사용해도 무방. 다만, 포팅 매뉴얼 기준은 bitnami/kafka 이다.
    - KRaft 모드 사용으로 zookeeper는 사용하지 않는다.

    - provectuslabs/kafka-ui
        - 0.7.2
- 도커 이미지 받기
    - bitnami/kafka

      ```
      sudo docker pull bitnami/kafka:latest
      ```

      ```
      sudo docker pull bitnami/kafka:3.7.0
      ```

    - provectuslabs/kafka-ui

      ```
      sudo docker pull provectuslabs/kafka-ui:latest
      ```

      ```
      sudo docker pull provectuslabs/kafka-ui:v0.7.2
      ```

- 도커 컨테이너 실행

  ```
  vi docker-compose-kafka.yaml
  ```

  ```
  # docker compose v2.25.0 이상에서는 version 명시 안 해도 됨
  # version: '3.9'

  networks:
    kafka_network:

  volumes:
    kafka00:
      driver: local
    kafka01:
      driver: local
    kafka02:
      driver: local

  services:
    kafka00service:
  ```

```
    image: bitnami/kafka:latest
    restart: unless-stopped
    container_name: kafka00container
    ports:
      - "10000:9094"
    environment:
      - KAFKA_CFG_BROKER_ID=0
      - KAFKA_CFG_NODE_ID=0
      - KAFKA_KRAFT_CLUSTER_ID=018f12cf-0f49-7716-928e-f
bb8dc811943
      - KAFKA_CFG_CONTROLLER_QUORUM_VOTERS=0@kafka00serv
ice:9093,1@kafka01service:9093,2@kafka02service:9093
      - ALLOW_PLAINTEXT_LISTENER=yes
      - KAFKA_CFG_AUTO_CREATE_TOPICS_ENABLE=true
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092,CONTROLLE
R://:9093,EXTERNAL://:9094
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka
00service:9092,EXTERNAL://k10a706.p.ssafy.io:10000
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CONTROL
LER:PLAINTEXT,EXTERNAL:PLAINTEXT,PLAINTEXT:PLAINTEXT
      - KAFKA_CFG_OFFSETS_TOPIC_REPLICATION_FACTOR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_REPLICATION_FACT
OR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_MIN_ISR=2
      - KAFKA_CFG_PROCESS_ROLES=controller,broker
      - KAFKA_CFG_CONTROLLER_LISTENER_NAMES=CONTROLLER
    networks:
      - kafka_network
    volumes:
      - kafka00:/bitnami/kafka

  kafka01service:
    image: bitnami/kafka:latest
    restart: unless-stopped
    container_name: kafka01container
    ports:
      - "10001:9094"
    environment:
```

```
      # 고유 식별자 설정
      - KAFKA_CFG_BROKER_ID=1
      - KAFKA_CFG_NODE_ID=1
      # 클러스터 설정
      - KAFKA_KRAFT_CLUSTER_ID=018f12cf-0f49-7716-928e-f
bb8dc811943
      # 컨트롤러 쿼럼 설정 (Raft Quorum을 구성하는 노드의 목록
정의)
      - KAFKA_CFG_CONTROLLER_QUORUM_VOTERS=0@kafka00serv
ice:9093,1@kafka01service:9093,2@kafka02service:9093
      # 평문 통신 허용 여부 설정
      - ALLOW_PLAINTEXT_LISTENER=yes
      # kafka broker가 자동으로 topic 생성 여부 설정
      - KAFKA_CFG_AUTO_CREATE_TOPICS_ENABLE=true
      # broker 사용 리스너 설정
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092,CONTROLLE
R://:9093,EXTERNAL://:9094
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka
01service:9092,EXTERNAL://k10a706.p.ssafy.io:10001
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CONTROL
LER:PLAINTEXT,EXTERNAL:PLAINTEXT,PLAINTEXT:PLAINTEXT
      # 트랜잭션, offset, 복제 factor, ISR 설정
      - KAFKA_CFG_OFFSETS_TOPIC_REPLICATION_FACTOR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_REPLICATION_FACT
OR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_MIN_ISR=2
      # broker 수행 역할 정의
      - KAFKA_CFG_PROCESS_ROLES=controller,broker
      - KAFKA_CFG_CONTROLLER_LISTENER_NAMES=CONTROLLER
    networks:
      - kafka_network
    volumes:
      - kafka01:/bitnami/kafka

  kafka02service:
    image: bitnami/kafka:latest
    restart: unless-stopped
    container_name: kafka02container
```

```
    ports:
      - "10002:9094"
    environment:
      # 고유 식별자 설정
      - KAFKA_CFG_BROKER_ID=2
      - KAFKA_CFG_NODE_ID=2
      # 클러스터 설정
      - KAFKA_KRAFT_CLUSTER_ID=018f12cf-0f49-7716-928e-f
bb8dc811943
      # 컨트롤러 쿼럼 설정 (Raft Quorum을 구성하는 노드의 목록
정의)
      - KAFKA_CFG_CONTROLLER_QUORUM_VOTERS=0@kafka00serv
ice:9093,1@kafka01service:9093,2@kafka02service:9093
      # 평문 통신 허용 여부 설정
      - ALLOW_PLAINTEXT_LISTENER=yes
      # kafka broker가 자동으로 topic 생성 여부 설정
      - KAFKA_CFG_AUTO_CREATE_TOPICS_ENABLE=true
      # broker 사용 리스너 설정
      - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092,CONTROLLE
R://:9093,EXTERNAL://:9094
      - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka
02service:9092,EXTERNAL://k10a706.p.ssafy.io:10002
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=CONTROL
LER:PLAINTEXT,EXTERNAL:PLAINTEXT,PLAINTEXT:PLAINTEXT
      # 트랜잭션, offset, 복제 factor, ISR 설정
      - KAFKA_CFG_OFFSETS_TOPIC_REPLICATION_FACTOR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_REPLICATION_FACT
OR=3
      - KAFKA_CFG_TRANSACTION_STATE_LOG_MIN_ISR=2
      # broker 수행 역할 정의
      - KAFKA_CFG_PROCESS_ROLES=controller,broker
      - KAFKA_CFG_CONTROLLER_LISTENER_NAMES=CONTROLLER
    networks:
      - kafka_network
    volumes:
      - kafka02:/bitnami/kafka

  kafka-web-ui:
```

```
    image: provectuslabs/kafka-ui:latest
    restart: always
    container_name: kafka-web-ui
    ports:
      - "8085:8080"
    environment:
      - KAFKA_CLUSTERS_0_NAME=Local-Kraft-Cluster
      - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka00servic
e:9092,kafka01service:9092,kafka02service:9092
      - DYNAMIC_CONFIG_ENABLED=true
      - KAFKA_CLUSTERS_0_AUDIT_TOPICAUDITENABLED=true
      - KAFKA_CLUSTERS_0_AUDIT_CONSOLEAUDITENABLED=true
    depends_on:
      - kafka00service
      - kafka01service
      - kafka02service
    networks:
      - kafka_network
```

```
docker compose -f docker-compose-kafka.yaml up -d
```

- 컨테이너 내부 접근 후 명령어 사용
  - `docker exec -it 카프카_컨테이명 /bin/bash` 로 내부 접근하면 `/opt/bitnami/kafka/bin` 에 명령 파일들이 위치해 있다.

RabbitMQ

- 버전 정보
  - 3.13.1-management
- 도커 이미지 받기

```
sudo docker pull rabbitmq:3.13.1-management
```

- 도커 컨테이너 실행

```
vi docker-compose-rabbitmq.yml
```

```
# docker-compose-rabbitmq.yml

services:
  rabbitmq:
    image: rabbitmq:3.13.1-management
    container_name: rabbitmq
    volumes:
      - ./rabbitmq-data/etc/:/etc/rabbitmq/
      - ./rabbitmq-data/data/:/var/lib/rabbitmq/
      - ./rabbitmq-data/logs/:/var/log/rabbitmq/
    ports:
      - "5672:5672" # AMQP
      - "15672:15672" # Management UI
      - "1883:1883" # MQTT
    environment:
      - RABBITMQ_ERLANG_COOKIE=018f2d94-1109-74a7-bfb6-0
aba298fe5ad
      - RABBITMQ_DEFAULT_USER=accio
      - RABBITMQ_DEFAULT_PASS=accio706
    restart: unless-stopped
```

```
docker compose -f docker-compose-rabbitmq.yml up -d
```

- Plugin 활성화

```
docker exec -it rabbitmq /bin/bash
```

```
rabbitmq-plugins enable rabbitmq_management
```

```
rabbitmq-plugins enable rabbitmq_mqtt
```

외부 서비스

Amazon S3

- S3 서비스 페이지로 이동한다. (검색 또는 https://s3.console.aws.amazon.com/s3)
- '버킷 만들기' 버튼을 클릭하여 버킷을 생성한다.
    - 일반 구성
        - AWS 리전과 버킷 이름을 설정.
    - 객체 소유권
        - 'ACL 비활성화됨' 선택.
    - 이 버킷의 퍼블릭 액세스 차단 설정
        - '새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단'만 설정하지 않음.
    - 버킷 버전 관리
        - '비활성화' 선택.
    - 기본 암호화
        - 암호화 유형
            - 'Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화' 선택.
        - 버킷 키
            - '활성화' 선택
    - 고급 설정
        - 객체 잠금
            - '비활성화' 선택
- S3 Access Key & Secret Key 발급
    - IAM 서비스 페이지로 이동한다. (검색 또는 https://us-east-1.console.aws.amazon.com/iam)
    - 좌측 메뉴 '액세스 관리 - 사용자'에서 사용자 생성을 진행한다.
        - 사용자 이름 입력 후 다음
        - 권한 옵션은 '직접 정책 연결' 선택하여 권한 정책에서 'AmazonS3FullAccess' 체크 후 다음
        - '사용자 생성' 버튼 클릭하여 생성
        - 사용자 목록에서 생성한 사용자 클릭하여 상세(관리) 페이지로 이동

        - ‘보안 자격 증명’ 탭에서 ‘액세스 키 만들기’ 클릭
            - ‘Command Line Interface(CLI)’ 선택하여 다음
            - 필요 시 태그를 설정하고 액세스 키 만들기
            - Access Key와 Secret Key를 확인 가능.
                - csv 파일을 다운받아 관리.
- 필요 시 버킷 정책을 추가 및 수정한다.
    - 경우에 따라 퍼블릭 액세스 차단 설정 및 IAM 사용자의 권한에 대해 추가 및 수정이 필요할 수도 있다.

환경 변수

Backend (Spring Boot)

application.yml

- Spring Boot 프로젝트 내 src/main/resources 에 위치

```
spring:
  datasource:
    url: # 데이터베이스 접속 URL
    username: # 데이터베이스 사용자 이름
    password: # 데이터베이스 사용자 비밀번호
    driver-class-name: com.mysql.cj.jdbc.Driver # JDBC 드라이버 클래스명
  jpa:
    hibernate:
      ddl-auto: validate # JPA 시작 시 데이터베이스와 엔티티 매핑 검증만 수행
    show-sql: true # 실행되는 SQL을 콘솔에서 보여줄지 여부
  servlet:
    multipart:
      max-file-size: 100MB # 최대 파일 크기 제한
      max-request-size: 100MB # 최대 요청 크기 제한
  rabbitmq:
    host: # RabbitMQ 서버 주소
```

```
    port: 5672 # RabbitMQ 서버 포트
    username: # RabbitMQ 사용자 이름
    password: # RabbitMQ 사용자 비밀번호
  kafka:
    bootstrap-servers: # Kafka 클러스터에 접속하기 위한 서버 목
록
    consumer:
      group-id: # Kafka 소비자 그룹 ID
      auto-offset-reset: earliest # 가장 초기의 오프셋부터 메시
지 처리 시작
      key-deserializer: org.apache.kafka.common.serializati
on.StringDeserializer # 메시지 키 역직렬화를 위한 클래스
      value-deserializer: org.apache.kafka.common.serializa
tion.StringDeserializer # 메시지 값 역직렬화를 위한 클래스
    producer:
      key-serializer: org.apache.kafka.common.serializatio
n.StringSerializer # 메시지 키 직렬화를 위한 클래스
      value-serializer: org.apache.kafka.common.serializati
on.StringSerializer # 메시지 값 직렬화를 위한 클래스

# RabbitMQ 설정
rabbitmq:
  queue:
    name: # RabbitMQ 큐 이름
    name2: # 두 번째 RabbitMQ 큐 이름
  exchange:
    name: # RabbitMQ 교환기 이름
  routing:
    key: # 라우팅 키 패턴 1
    key2: # 라우팅 키 패턴 2

# S3 설정
cloud:
  aws:
    s3:
      bucket: # S3 버킷 이름
    credentials:
      access-key: # AWS 액세스 키
```

```
      secret-key: # AWS 비밀 키
    region:
      static: ap-northeast-2 # AWS 리전
    stack:
      auto: false # AWS 스택 자동 생성 여부
```

Frontend (Vue.js / WEB)

.env

- Vue.js 프로젝트 내 루트에 위치

```
VUE_APP_KAKAO_MAP_API_KEY= # 카카오 지도 API를 사용하기 위한 키. 해당 키를 사용하여 카카오 지도 서비스에 접근 가능.
VUE_APP_API_URL= # 애플리케이션에서 사용하는 백엔드 API의 기본 URL. 이 URL을 통해 백엔드 서버의 API에 접근.
```

CI/CD

- Frontend는 Pipeline script from SCM 사용.
  - 각 Jekins Pipeline에 대해 groovy 파일 작성.

Backend (Spring Boot)

Dockerfile

```
# docker buildx 이용
FROM docker
COPY --from=docker/buildx-bin:latest /buildx /usr/libexec/docker/cli-plugins/docker-buildx

# 기본 이미지로 eclipse temurin 기반 17 jdk 사용
FROM eclipse-temurin:17-jdk
# 애플리케이션 JAR 파일 복사
ADD ./build/libs/isegye-0.0.1-SNAPSHOT.jar app.jar
```

```
# 실행할 명령어 설정
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Jenkins Pipeline

```
def releasePort
def containerName

pipeline{
    agent any

    environment {
        imageName = "demise1426/accio-isegye-be" // docker
hub의 이미지 이름
        registryCredential = 'demise1426-docker' // docker
hub access token

        gitBranch = 'develop-be'
        gitCredential = 'demise1426-gitlab-sub'
        gitUrl = 'https://lab.ssafy.com/s10-final/S10P31A70
6.git'

        releaseServerAccount = 'ubuntu' // ssh 연결 시 사용할
user
        releaseServerUri = 'k10a706.p.ssafy.io' // 서비스 ur
l
        containerPort = '8080' // 컨테이너 포트
        bluePort = '8080' // blue포트
        greenPort = '8082' // green포트

        MATTERMOST_ENDPOINT = credentials('mattermost_endpo
int')
        MATTERMOST_CHANNEL = credentials('mattermost_channe
l')
    }

    stages {
        stage('Git Clone') {
```

```
            steps {
                git branch: gitBranch,
                    credentialsId: gitCredential,
                    url: gitUrl
            }
        }

        stage('Jar Build') {
            steps {
                dir('backend') {
                    withCredentials([file(credentialsId: 'A
PPLICATION_YML', variable: 'APPLICATION_YML')]) {
                        withEnv(["MY_APPLICATION_YML=${APPL
ICATION_YML}"]) {
                            sh 'mkdir -p ./src/main/resourc
es && cp ${MY_APPLICATION_YML} ./src/main/resources/applica
tion.yml'
                        }
                    }
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean bootJar'
                }
            }
        }

        stage('Docker Image Build & DockerHub Push') {
            steps {
                dir('backend') {
                    script {
                        // Docker Hub에 로그인 (Docker Hub A
ccess Token이 필요)
                        docker.withRegistry('', registryCre
dential) {
                            sh "docker buildx create --use
--name mybuilder"
                            sh "docker buildx build --platf
orm linux/amd64,linux/arm64 -t $imageName:$BUILD_NUMBER -t
$imageName:latest --push ."
```

```
                        }

                    }
                }
            }
        }

        stage('Blue/Green Port Check') { // 서비스 중단 전 기존 컨테이너 및 이미지 삭제
            steps {
                script {
                    // curl 명령어의 결과를 확인하여 포트 번호를 결정합니다.
                    def isBlueUp = sh(script: "curl -s --fail http://${releaseServerUri}:${bluePort}", returnStatus: true) == 0
                    if (isBlueUp) {
                        releasePort = greenPort
                        containerName = 'accio-isegye-be_g'
                    } else {
                        releasePort = bluePort
                        containerName = 'accio-isegye-be_b'
                    }
                    echo "isBlueUp : $isBlueUp, Port selected: $releasePort, container name: $containerName"
                }
            }
        }

        stage('DockerHub Pull') { // docker hub에서 이미지 pull
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh "ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri 'sudo docker pull $imageName:latest'"
                }
            }
```

```
        }

        stage('Service Start') { // pull된 이미지 이용하여 doc
ker 컨테이너 실행
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh """
                    echo "port : ${releasePort}, container
: ${containerName}"
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "sudo docker run -i -e
TZ=Asia/Seoul --name ${containerName} -p ${releasePort}:${c
ontainerPort} -d ${imageName}:latest"
                    """
                }
            }
        }

        stage('Switch Nginx Port & Nginx reload') { //NginX
Port 변경
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh """
                    ssh -o StrictHostKeyChecking=no $releas
eServerAccount@$releaseServerUri "echo 'set \\\$be_service_
url http://${releaseServerUri}:${releasePort};' | sudo tee
/home/ubuntu/data/nginx/conf.d/be-service-url.inc > /dev/nu
ll && sudo docker exec nginx nginx -s reload"
                    echo "Switch the reverse proxy directio
n of nginx to ${releasePort} 🔄"
                    """
                }
            }
        }



        stage('Service Check & Kill the Old Container') {
// 연결 체크 & 예전 컨테이너 삭제
```

```
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    script {
                        def retry_count = 0
                        for (retry_count = 0; retry_count <
20; retry_count++) {
                            def isRunning = sh(script: "cur
l -s --fail http://${releaseServerUri}:${releasePort}/", re
turnStatus: true) == 0
                            if (isRunning) {
                                if(releasePort==bluePort){
                                    sh "ssh -o StrictHostKe
yChecking=no $releaseServerAccount@$releaseServerUri 'docke
r rm accio-isegye-be_g -f'"
                                }else{
                                    sh "ssh -o StrictHostKe
yChecking=no $releaseServerAccount@$releaseServerUri 'docke
r rm accio-isegye-be_b -f'"
                                }
                                echo "Killed the process on
the opposite server."
                                sh "ssh -o StrictHostKeyChe
cking=no $releaseServerAccount@$releaseServerUri 'docker im
age prune -f'"
                                break
                            } else {
                                if (retry_count == 19) {
                                    error("The server is no
t alive after 20 attempts. Exiting...")
                                }
                                echo "The server is not ali
ve yet. Retry health check in 5 seconds..."
                                sleep 5
                            }
                        }
                    }
                }
            }
```

```
        }
    }

        post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (
                    color: 'good',
                    message: "Backend Build Success: ${env.
JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Nam
e})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (
                    color: 'danger',
                    message: "Backend Build Failure: ${env.
JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Nam
e})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
    }
```

```
}
```

Frontend (Vue.js / Web)

Dockerfile

```
FROM docker
COPY --from=docker/buildx-bin:latest /buildx /usr/libexec/d
ocker/cli-plugins/docker-buildx

# node 21 alpine 이미지를 기반으로 설정
FROM node:21-alpine

# 작업 디렉토리를 /app으로 설정
WORKDIR /app

# package.json 및 package-lock.json 파일을 /app 디렉토리로 복사
COPY package*.json ./

# update npm version
RUN npm install -g npm@10.8.0

# 프로젝트 의존성 설치
RUN npm install

# .env 파일을 /app 디렉토리로 복사
COPY .env .

# 로컬 시스템의 현재 디렉토리에서 /app 디렉토리로 복사
COPY . .

# Vue.js 애플리케이션 Build
RUN npm run build

# 3000 포트를 외부에 노출
EXPOSE 3000
```

```
# Vue.js 애플리케이션 실행
CMD ["npm", "run", "serve"]
```

Jenkins Pipeline

```
def releasePort
def containerName

pipeline {
    agent any
    tools {
        nodejs "nodejs21.5.0"
    }

    environment {
        webPath = 'frontend/Web/boardgame'

        gitBranch = 'develop-fe'
        gitCredential = 'demise1426-gitlab-sub'
        gitUrl = 'https://lab.ssafy.com/s10-final/S10P31A70
6.git'

        imageName = "demise1426/accio-isegye-web" // docker
hub의 이미지 이름
        registryCredential = 'demise1426-docker' // docker
hub access token

        releaseServerAccount = 'ubuntu' // ssh 연결 시 사용할
user
        releaseServerUri = 'k10a706.p.ssafy.io' // 서비스 ur
l
        containerPort = '3000' // 컨테이너 포트
        bluePort = '3000' // blue포트
        greenPort = '3001' // green포트

        MATTERMOST_ENDPOINT = credentials('mattermost_endpo
int')
        MATTERMOST_CHANNEL = credentials('mattermost_channe
```

```
l')
    }

    stages {
        stage('Check Changes') {
            steps {
                script {
                    // GitLab webhook payload contains info
rmation about the changes
                    def changes = currentBuild.rawBuild.cha
ngeSets.collect { changeLogSet ->
                        changeLogSet.collect { changeSet ->
                            changeSet.getAffectedFiles()
                        }
                    }.flatten()

                    // Check if changes include web boardga
me directory
                    def webChanged = changes.any { it.path.
startsWith(webPath) }

                    if (webChanged) {
                        echo 'Changes detected in frontend/
Web/boardgame directory. Running the pipeline.'
                    } else {
                        echo 'No changes in frontend/Web/bo
ardgame directory. Skipping the pipeline.'
                        currentBuild.result = 'ABORTED'
                        error 'No changes in frontend/Web/b
oardgame directory. Skipping the pipeline.'
                    }
                }
            }
        }

        stage('Git Clone') {
            steps {
                git branch: gitBranch,
```

```
                        credentialsId: gitCredential,
                        url: gitUrl
            }
        }

        stage('Node Build') {
            steps {
                dir('frontend/Web/boardgame') {
                    withCredentials([file(credentialsId: 'W
EB_ENV', variable: 'WEB_ENV')]) {
                        withEnv(["MY_WEB_ENV=${WEB_ENV}"])
{
                            sh 'cp ${MY_WEB_ENV} .env'
                        }
                    }
                    sh 'npm install'
                    sh 'npm run build'
                }
            }
        }

        stage('Docker Image Build & DockerHub Push') {
            steps {
                dir('frontend/Web/boardgame') {
                    script {
                        docker.withRegistry('', registryCre
dential) {
                            sh "docker buildx create --use
--name mybuilder"
                            sh "docker buildx build --platf
orm linux/amd64,linux/arm64 -t $imageName:$BUILD_NUMBER -t
$imageName:latest --push ."
                        }
                    }
                }
            }
        }
```

```
        stage('Blue/Green Port Check') { // 서비스 중단 전 기존 컨테이너 및 이미지 삭제
            steps {
                script {
                    // curl 명령어의 결과를 확인하여 포트 번호를 결정합니다.
                    def isBlueUp = sh(script: "curl -s --fail http://${releaseServerUri}:${bluePort}", returnStatus: true) == 0
                    if (isBlueUp) {
                        releasePort = greenPort
                        containerName = 'accio-isegye-web_g'
                    } else {
                        releasePort = bluePort
                        containerName = 'accio-isegye-web_b'
                    }
                    echo "isBlueUp : $isBlueUp, Port selected: $releasePort, container name: $containerName"
                }
            }
        }

        stage('DockerHub Pull') { // docker hub에서 이미지 pull
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh "ssh -o StrictHostKeyChecking=no $releaseServerAccount@$releaseServerUri 'sudo docker pull $imageName:latest'"
                }
            }
        }

        stage('Service Start') { // pull된 이미지 이용하여 docker 컨테이너 실행
            steps {
```

```
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh """
                    echo "port : ${releasePort}, container
: ${containerName}"
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "sudo docker run -i -e
TZ=Asia/Seoul --name ${containerName} -p ${releasePort}:${c
ontainerPort} -d ${imageName}:latest"
                    """
                }
            }
        }

        stage('Switch Nginx Port & Nginx reload') { //NginX
Port 변경
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh """
                    ssh -o StrictHostKeyChecking=no $releas
eServerAccount@$releaseServerUri "echo 'set \\\$fe_service_
url http://${releaseServerUri}:${releasePort};' | sudo tee
/home/ubuntu/data/nginx/conf.d/fe-service-url.inc > /dev/nu
ll && sudo docker exec nginx nginx -s reload"
                    echo "Switch the reverse proxy directio
n of nginx to ${releasePort} 🔄"
                    """
                }
            }
        }

        stage('Service Check & Kill the Old Container') {
// 연결 체크 & 예전 컨테이너 삭제
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    script {
                        def retry_count = 0
                        for (retry_count = 0; retry_count <
20; retry_count++) {
```

```
                            def isRunning = sh(script: "cur
l -s --fail http://${releaseServerUri}:${releasePort}/", re
turnStatus: true) == 0
                            if (isRunning) {
                                if(releasePort==bluePort){
                                    sh "ssh -o StrictHostKe
yChecking=no $releaseServerAccount@$releaseServerUri 'docke
r rm accio-isegye-web_g -f'"
                                }else{
                                    sh "ssh -o StrictHostKe
yChecking=no $releaseServerAccount@$releaseServerUri 'docke
r rm accio-isegye-web_b -f'"
                                }
                                echo "Killed the process on
the opposite server."
                                sh "ssh -o StrictHostKeyChe
cking=no $releaseServerAccount@$releaseServerUri 'docker im
age prune -f'"
                                break
                            } else {
                                if (retry_count == 19) {
                                    error("The server is no
t alive after 20 attempts. Exiting...")
                                }
                                echo "The server is not ali
ve yet. Retry health check in 5 seconds..."
                                sleep 5
                            }
                        }
                    }
                }
            }
        }
    }

    post {
        success {
            script {
```

```
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (
                    color: 'good',
                    message: "Fe-Web Build Success: ${env.J
OB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Nam
e})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (
                    color: 'danger',
                    message: "Fe-Web Build Failure: ${env.J
OB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Nam
e})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
    }
}
```

Frontend (Android App / Manager)

Dockerfile

```
FROM docker
COPY --from=docker/buildx-bin:latest /buildx /usr/libexec/d
ocker/cli-plugins/docker-buildx

FROM eclipse-temurin:17-jdk

ENV ANDROID_COMPILE_SDK="34" \
    ANDROID_SDK_BUILD_TOOLS="34.0.0" \
    ANDROID_SDK_PLATFORM_TOOLS="35.0.1" \
    ANDROID_SDK_COMMANDLINE_TOOLS="11076708"

RUN apt-get update && \
    apt-get install -y curl tar bash procps unzip && \
    rm -rf /var/lib/apt/lists/*

WORKDIR /android-sdk

RUN curl -o cmdline-tools.zip "https://dl.google.com/androi
d/repository/commandlinetools-linux-${ANDROID_SDK_COMMANDLI
NE_TOOLS}_latest.zip" && \
    unzip cmdline-tools.zip -d . && \
    rm cmdline-tools.zip && \
    mv cmdline-tools cmdline-tools-temp && \
    mkdir -p cmdline-tools/latest && \
    mv cmdline-tools-temp/* cmdline-tools/latest/ && \
    rm -r cmdline-tools-temp

ENV ANDROID_HOME=/android-sdk
ENV PATH $PATH:$ANDROID_HOME/cmdline-tools/latest/bin

RUN yes | sdkmanager --licenses

RUN sdkmanager "platform-tools" "platforms;android-${ANDROI
D_COMPILE_SDK}" "build-tools;${ANDROID_SDK_BUILD_TOOLS}"

WORKDIR /app

COPY . .
```

```
RUN chmod +x gradlew && \
    # ./gradlew assembleRelease
    ./gradlew assembleDebug
# app/build/outputs/apk/release/app-release.apk 생성
# app/build/outputs/apk/debug/app-debug.apk 생성

ENTRYPOINT ["tail", "-f", "/dev/null"]
```

Jenkins Pipeline

```
def s3url

pipeline {
    agent any

    environment {
        targetPath = 'frontend/BgManager'

        gitBranch = 'develop-fe'
        gitCredential = 'demise1426-gitlab-sub'
        gitUrl = 'https://lab.ssafy.com/s10-final/S10P31A70
6.git'

        imageName = "demise1426/accio-isegye-android-manage
r" // docker hub의 이미지 이름
        registryCredential = 'demise1426-docker' // docker
hub access token

        apkFileContainerPath = '/app/app/build/outputs/apk/
debug/app-debug.apk'
        apkFileLocalPath = '/home/ubuntu/apk/manager/'
        apkS3Path = 'apk/manager/'
        s3BucketName = 'accio-isegye'
        awsRegion = 'ap-northeast-2'

        releaseServerAccount = 'ubuntu' // ssh 연결 시 사용할
```

```
user
        releaseServerUri = 'k10a706.p.ssafy.io' // 서비스 ur
l
        containerName = 'accio-isegye-android-manager'

        MATTERMOST_ENDPOINT = credentials('mattermost_endpo
int')
        MATTERMOST_CHANNEL = credentials('mattermost_channe
l')
    }

    stages {
        stage('Check Changes') {
            steps {
                script {
                    // GitLab webhook payload contains info
rmation about the changes
                    def changes = currentBuild.rawBuild.cha
ngeSets.collect { changeLogSet ->
                        changeLogSet.collect { changeSet ->
                            changeSet.getAffectedFiles()
                        }
                    }.flatten()

                    // Check if changes include frontend ma
nager app directory
                    def targetChanged = changes.any { it.pa
th.startsWith(targetPath) }

                    if (targetChanged) {
                        echo 'Changes detected in frontend/
BgManager directory. Running the pipeline.'
                    } else {
                        echo 'No changes in frontend/BgMana
ger directory. Skipping the pipeline.'
                        currentBuild.result = 'ABORTED'
                        error 'No changes in frontend/BgMan
ager directory. Skipping the pipeline.'
```

```
                    }
                }
            }
        }

        stage('Git Clone') {
            steps {
                git branch: gitBranch,
                        credentialsId: gitCredential,
                        url: gitUrl
            }
        }

        stage('Docker Image Build & DockerHub Push') {
            steps {
                dir('frontend/BgManager') {
                    script {
                        docker.withRegistry('', registryCre
dential) {
                            sh "docker buildx create --use
--name mybuilder"
                            sh "docker buildx build --platf
orm linux/amd64 -t $imageName:$BUILD_NUMBER -t $imageName:l
atest --push ."
                        }
                    }
                }
            }
        }

        stage('DockerHub Pull') { // docker hub에서 이미지 pu
ll
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh "ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri 'sudo docker pull $ima
geName:latest'"
                }
```

```
            }
        }

        stage('Docker Run & File Copy') {
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh """
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "sudo docker run --nam
e ${containerName} -d ${imageName}:latest"
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "rm -f ${apkFileLocalP
ath}app-debug.apk"
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "sudo docker cp ${cont
ainerName}:${apkFileContainerPath} ${apkFileLocalPath}"
                        scp -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri:${apkFileLocalPath}app
-debug.apk .
                    """
                }
            }
        }

        stage('S3 Upload') {
            steps {
                withAWS(credentials:'AWS-IAM') {
                    s3Upload(file:'./app-debug.apk', bucke
t:"${s3BucketName}", path:"${apkS3Path}${BUILD_NUMBER}/app-
debug.apk")
                }
                script {
                    s3url = "https://${s3BucketName}.s3.${a
wsRegion}.amazonaws.com/${apkS3Path}${BUILD_NUMBER}/app-deb
ug.apk"
                }
            }
        }
```

```
        stage('Docker stop & rm & rmi') {
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh "docker stop ${containerName}"
                    sh "docker rm ${containerName}"
                    sh "docker rmi ${imageName}:latest"
                }
            }
        }

    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (
                    color: 'good',
                    message: "Fe-Manager App Build Success:
${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Auth
or_Name})\nAPK FILE S3 LINK:${s3url}\n(<${env.BUILD_URL}|De
tails>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (
```

```
                    color: 'danger',
                    message: "Fe-Manager App Build Failure:
${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Auth
or_Name})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
    }
}
```

Frontend (Android App / Customer)

Dockerfile

```
FROM docker
COPY --from=docker/buildx-bin:latest /buildx /usr/libexec/d
ocker/cli-plugins/docker-buildx

FROM eclipse-temurin:17-jdk

ENV ANDROID_COMPILE_SDK="34" \
    ANDROID_SDK_BUILD_TOOLS="34.0.0" \
    ANDROID_SDK_PLATFORM_TOOLS="35.0.1" \
    ANDROID_SDK_COMMANDLINE_TOOLS="11076708"

RUN apt-get update && \
    apt-get install -y curl tar bash procps unzip && \
    rm -rf /var/lib/apt/lists/*

WORKDIR /android-sdk

RUN curl -o cmdline-tools.zip "https://dl.google.com/androi
d/repository/commandlinetools-linux-${ANDROID_SDK_COMMANDLI
NE_TOOLS}_latest.zip" && \
    unzip cmdline-tools.zip -d . && \
    rm cmdline-tools.zip && \
```

```
    mv cmdline-tools cmdline-tools-temp && \
    mkdir -p cmdline-tools/latest && \
    mv cmdline-tools-temp/* cmdline-tools/latest/ && \
    rm -r cmdline-tools-temp

ENV ANDROID_HOME=/android-sdk
ENV PATH $PATH:$ANDROID_HOME/cmdline-tools/latest/bin

RUN yes | sdkmanager --licenses

RUN sdkmanager "platform-tools" "platforms;android-${ANDROI
D_COMPILE_SDK}" "build-tools;${ANDROID_SDK_BUILD_TOOLS}"

WORKDIR /app

COPY . .


RUN chmod +x gradlew && \
    # ./gradlew assembleRelease
    ./gradlew assembleDebug
# app/build/outputs/apk/release/app-release.apk 생성(서명 필
요)
# app/build/outputs/apk/debug/app-debug.apk 생성

# 컨테이너를 실행 상태로 유지하기 위해 로그 확인
ENTRYPOINT ["tail", "-f", "/dev/null"]
```

Jenkins Pipeline

```
def s3url

pipeline {
    agent any

    environment {
        targetPath = 'frontend/IsegyeBoard'
```

```
        gitBranch = 'develop-fe'
        gitCredential = 'demise1426-gitlab-sub'
        gitUrl = 'https://lab.ssafy.com/s10-final/S10P31A70
6.git'

        imageName = "demise1426/accio-isegye-android-custom
er" // docker hub의 이미지 이름
        registryCredential = 'demise1426-docker' // docker
hub access token

        apkFileContainerPath = '/app/app/build/outputs/apk/
debug/app-debug.apk'
        apkFileLocalPath = '/home/ubuntu/apk/customer/'
        apkS3Path = 'apk/customer/'
        s3BucketName = 'accio-isegye'
        awsRegion = 'ap-northeast-2'

        releaseServerAccount = 'ubuntu' // ssh 연결 시 사용할
user
        releaseServerUri = 'k10a706.p.ssafy.io' // 서비스 ur
l
        containerName = 'accio-isegye-android-customer'

        MATTERMOST_ENDPOINT = credentials('mattermost_endpo
int')
        MATTERMOST_CHANNEL = credentials('mattermost_channe
l')
    }

    stages {
        stage('Check Changes') {
            steps {
                script {
                    // GitLab webhook payload contains info
rmation about the changes
                    def changes = currentBuild.rawBuild.cha
ngeSets.collect { changeLogSet ->
                        changeLogSet.collect { changeSet ->
```

```
                            changeSet.getAffectedFiles()
                        }
                    }.flatten()

                    // Check if changes include frontend cu
stomer app directory
                    def targetChanged = changes.any { it.pa
th.startsWith(targetPath) }

                    if (targetChanged) {
                        echo 'Changes detected in frontend/
IsegyeBoard directory. Running the pipeline.'
                    } else {
                        echo 'No changes in frontend/Isegye
Board directory. Skipping the pipeline.'
                        currentBuild.result = 'ABORTED'
                        error 'No changes in frontend/Isegy
eBoard directory. Skipping the pipeline.'
                    }
                }
            }
        }

        stage('Git Clone') {
            steps {
                git branch: gitBranch,
                        credentialsId: gitCredential,
                        url: gitUrl
            }
        }

        stage('Docker Image Build & DockerHub Push') {
            steps {
                dir('frontend/IsegyeBoard') {
                    script {
                        docker.withRegistry('', registryCre
dential) {
                            sh "docker buildx create --use
```

```
--name mybuilder"
                            sh "docker buildx build --platf
orm linux/amd64 -t $imageName:$BUILD_NUMBER -t $imageName:l
atest --push ."
                        }
                    }
                }
            }
        }

        stage('DockerHub Pull') { // docker hub에서 이미지 pu
ll
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh "ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri 'sudo docker pull $ima
geName:latest'"
                }
            }
        }

        stage('Docker Run & File Copy') {
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh """
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "sudo docker run --nam
e ${containerName} -d ${imageName}:latest"
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "rm -f ${apkFileLocalP
ath}app-debug.apk"
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "sudo docker cp ${cont
ainerName}:${apkFileContainerPath} ${apkFileLocalPath}"
                        scp -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri:${apkFileLocalPath}app
-debug.apk .
                    """
```

```
                }
            }
        }

        stage('S3 Upload') {
            steps {
                withAWS(credentials:'AWS-IAM') {
                    s3Upload(file:'./app-debug.apk', bucke
t:"${s3BucketName}", path:"${apkS3Path}${BUILD_NUMBER}/app-
debug.apk")
                }
                script {
                    s3url = "https://${s3BucketName}.s3.${a
wsRegion}.amazonaws.com/${apkS3Path}${BUILD_NUMBER}/app-deb
ug.apk"
                }
            }
        }

        stage('Docker stop & rm & rmi') {
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh "docker stop ${containerName}"
                    sh "docker rm ${containerName}"
                    sh "docker rmi ${imageName}:latest"
                }
            }
        }

    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
```

```
                mattermostSend (
                    color: 'good',
                    message: "Fe-Customer App Build Succes
s: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${A
uthor_Name})\nAPK FILE S3 LINK:${s3url}\n(<${env.BUILD_URL}
|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (
                    color: 'danger',
                    message: "Fe-Customer App Build Failur
e: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${A
uthor_Name})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
    }
}
```

AI (FastAPI)

requirements.txt

```
albumentations==1.4.7
annotated-types==0.6.0
anyio==4.3.0
certifi==2024.2.2
```

```
charset-normalizer==3.3.2
click==8.1.7
colorama==0.4.6
coloredlogs==15.0.1
contourpy==1.2.1
cycler==0.12.1
Cython==3.0.10
dnspython==2.6.1
easydict==1.13
email_validator==2.1.1
exceptiongroup==1.2.1
fastapi==0.111.0
fastapi-cli==0.0.3
flatbuffers==24.3.25
fonttools==4.51.0
h11==0.14.0
httpcore==1.0.5
httptools==0.6.1
httpx==0.27.0
humanfriendly==10.0
idna==3.7
imageio==2.34.1
insightface==0.7.3
Jinja2==3.1.4
joblib==1.4.2
kiwisolver==1.4.5
lazy_loader==0.4
markdown-it-py==3.0.0
MarkupSafe==2.1.5
matplotlib==3.8.4
mdurl==0.1.2
mpmath==1.3.0
networkx==3.3
numpy==1.26.4
onnx==1.16.0
onnxruntime==1.17.3
opencv-python==4.9.0.80
opencv-python-headless==4.9.0.80
```

```
orjson==3.10.3
packaging==24.0
pandas==2.2.2
pillow==10.3.0
prettytable==3.10.0
protobuf==5.26.1
pydantic==2.7.1
pydantic_core==2.18.2
Pygments==2.18.0
pyparsing==3.1.2
pyreadline3==3.4.1
python-dateutil==2.9.0.post0
python-dotenv==1.0.1
python-multipart==0.0.9
pytz==2024.1
PyYAML==6.0.1
requests==2.31.0
rich==13.7.1
scikit-image==0.23.2
scikit-learn==1.4.2
scipy==1.13.0
shellingham==1.5.4
six==1.16.0
sniffio==1.3.1
starlette==0.37.2
sympy==1.12
threadpoolctl==3.5.0
tifffile==2024.5.10
tqdm==4.66.4
typer==0.12.3
typing_extensions==4.11.0
tzdata==2024.1
ujson==5.10.0
urllib3==2.2.1
uvicorn==0.29.0
watchfiles==0.21.0
wcwidth==0.2.13
websockets==12.0
```

Dockerfile

```
FROM docker
COPY --from=docker/buildx-bin:latest /buildx /usr/libexec/d
ocker/cli-plugins/docker-buildx

# 파이썬 3.10.6 버전을 기본 이미지로 사용합니다.
FROM python:3.10.6

# 작업 디렉토리 설정
WORKDIR /app

# 시스템 레벨 의존성 설치
RUN apt-get update && apt-get install -y \
    build-essential \
    libssl-dev \
    libffi-dev \
    python3-dev \
    && rm -rf /var/lib/apt/lists/*

# pip를 최신 버전으로 업그레이드
RUN pip install --upgrade pip

# 의존성 파일을 먼저 복사
COPY requirements.txt .

# 의존성 설치
RUN pip install --no-cache-dir --upgrade -r requirements.tx
t

# 나머지 파일 복사
COPY . .

# 호스트와 연결할 포트 번호
EXPOSE 8000

# FastAPI 서버 실행
CMD ["fastapi", "run", "main.py", "--port", "8000"]
```

Jenkins Pipeline

```
def releasePort
def containerName

pipeline{
    agent any

    environment {
        imageName = "demise1426/accio-isegye-ai" // docker
hub의 이미지 이름
        registryCredential = 'demise1426-docker' // docker
hub access token

        gitBranch = 'develop-ai'
        gitCredential = 'demise1426-gitlab-sub'
        gitUrl = 'https://lab.ssafy.com/s10-final/S10P31A70
6.git'

        releaseServerAccount = 'ubuntu' // ssh 연결 시 사용할
user
        releaseServerUri = 'k10a706.p.ssafy.io' // 서비스 ur
l
        containerPort = '8000' // 컨테이너 포트
        bluePort = '8000' // blue포트
        greenPort = '8001' // green포트

        MATTERMOST_ENDPOINT = credentials('mattermost_endpo
int')
        MATTERMOST_CHANNEL = credentials('mattermost_channe
l')
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: gitBranch,
                    credentialsId: gitCredential,
```

```
                    url: gitUrl
            }
        }

        stage('Docker Image Build & DockerHub Push') {
            steps {
                dir('ai') {
                    script {
                        // Docker Hub에 로그인 (Docker Hub A
ccess Token이 필요)
                        docker.withRegistry('', registryCre
dential) {
                            sh "docker buildx create --use
--name mybuilder"
                            sh "docker buildx build --platf
orm linux/amd64,linux/arm64 -t $imageName:$BUILD_NUMBER -t
$imageName:latest --push ."
                        }

                    }
                }
            }
        }

        stage('Blue/Green Port Check') { // 서비스 중단 전 기
존 컨테이너 및 이미지 삭제
            steps {
                script {
                    // curl 명령어의 결과를 확인하여 포트 번호를
결정합니다.
                    def isBlueUp = sh(script: "curl -s --fa
il http://${releaseServerUri}:${bluePort}/health", returnSt
atus: true) == 0
                    if (isBlueUp) {
                        releasePort = greenPort
                        containerName = 'accio-isegye-ai_g'
                    } else {
                        releasePort = bluePort
```

```
                        containerName = 'accio-isegye-ai_b'
                    }
                    echo "isBlueUp : $isBlueUp, Port select
ed: $releasePort, container name: $containerName"
                }
            }
        }

        stage('DockerHub Pull') { // docker hub에서 이미지 pu
ll
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh "ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri 'sudo docker pull $ima
geName:latest'"
                }
            }
        }

        stage('Service Start') { // pull된 이미지 이용하여 doc
ker 컨테이너 실행
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh """
                    echo "port : ${releasePort}, container
: ${containerName}"
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "sudo docker run -i -e
TZ=Asia/Seoul --name ${containerName} -p ${releasePort}:${c
ontainerPort} -d ${imageName}:latest"
                    """
                }
            }
        }

        stage('Switch Nginx Port & Nginx reload') { //NginX
Port 변경
            steps {
```

```
                sshagent(credentials: ['SSH-ubuntu']) {
                    sh """
                    ssh -o StrictHostKeyChecking=no $releas
eServerAccount@$releaseServerUri "echo 'set \\\$ai_service_
url http://${releaseServerUri}:${releasePort};' | sudo tee
/home/ubuntu/data/nginx/conf.d/ai-service-url.inc > /dev/nu
ll && sudo docker exec nginx nginx -s reload"
                    echo "Switch the reverse proxy directio
n of nginx to ${releasePort} 🔄"
                    """
                }
            }
        }


        stage('Service Check & Kill the Old Container') {
// 연결 체크 & 예전 컨테이너 삭제
            steps {
                sshagent(credentials: ['SSH-ubuntu']) {
                    script {
                        def retry_count = 0
                        for (retry_count = 0; retry_count <
20; retry_count++) {
                            def isRunning = sh(script: "cur
l -s --fail http://${releaseServerUri}:${releasePort}/healt
h", returnStatus: true) == 0
                            if (isRunning) {
                                if(releasePort==bluePort){
                                    sh "ssh -o StrictHostKe
yChecking=no $releaseServerAccount@$releaseServerUri 'docke
r rm accio-isegye-ai_g -f'"
                                }else{
                                    sh "ssh -o StrictHostKe
yChecking=no $releaseServerAccount@$releaseServerUri 'docke
r rm accio-isegye-ai_b -f'"
                                }
                                echo "Killed the process on
the opposite server."
```

```
                                    sh "ssh -o StrictHostKeyChe
cking=no $releaseServerAccount@$releaseServerUri 'docker im
age prune -f'"
                                    break
                                } else {
                                    if (retry_count == 19) {
                                        error("The server is no
t alive after 20 attempts. Exiting...")
                                    }
                                    echo "The server is not ali
ve yet. Retry health check in 5 seconds..."
                                    sleep 5
                                }
                            }
                        }
                    }
                }
            }
        }

        post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (
                    color: 'good',
                    message: "AI Server Build Success: ${en
v.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_N
ame})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
        failure {
```

```
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (
                    color: 'danger',
                    message: "AI Server Build Failure: ${en
v.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_N
ame})\n(<${env.BUILD_URL}|Details>)",
                    endpoint: MATTERMOST_ENDPOINT,
                    channel: MATTERMOST_CHANNEL
                )
            }
        }
    }


}
```

Embedded 환경

Ubuntu 20.04 LTS ROS2

```
ros2 설치 링크
https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Debi

[Set locale]
따라 하기

[Setup Sources]
따라 하기

[Install ROS2 packages]
중간에
- sudo apt install ros-foxy-desktop python3-argcomplete
```

```
- sudo apt install ros-foxy-ros-base python3-argcomplete
두 개 있는데, 위에 명령어로 다운로드(ROS2 전체 다운로드), 아래는 lite

[error]
sudo apt update
- E: Unable to locate package ros-foxy-desktop
sudo apt install ros-foxy-ros-base python3-argcomplete
-The repository 'http://packages.ros.org/ros2/ubuntu focal In
=> solution :
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistr
```

```
ROS2 개발 tools 설치

sudo apt update && sudo apt install -y \
  build-essential \
  cmake \
  git \
  libbullet-dev \
  python3-colcon-common-extensions \
  python3-flake8 \
  python3-pip \
  python3-pytest-cov \
  python3-rosdep \
  python3-setuptools \
  python3-vcstool \
  wget

# install some pip packages needed for testing

python3 -m pip install -U \
  argcomplete \
  flake8-blind-except \
  flake8-builtins \
  flake8-class-newline \
  flake8-comprehensions \
  flake8-deprecated \
  flake8-docstrings \
  flake8-import-order \
```

```
  flake8-quotes \
  pytest-repeat \
  pytest-rerunfailures \
  pytest

# install Fast-RTPS dependencies

sudo apt install --no-install-recommends -y \
  libasio-dev \
  libtinyxml2-dev

# install Cyclone DDS dependencies

sudo apt install --no-install-recommends -y \
  libcunit1-dev

ROS2 명령어 자동 완성
sudo apt install -y python3-argcomplete

[참고]
https://velog.io/@ssw9999/ROS2-Installation
https://intuitive-robotics.tistory.com/175

ROS2 Build Test
$ source /opt/ros/foxy/setup.bash (call C:\dev\ros2-eloquent\
$ mkdir -p ~/robot_ws/src
$ cd robot_ws/
$ colcon build --symlink-install

src 이동 후 build type에 맞는 패키지 생성
ros2 pkg create --build-type ament_cmake <my_package_name>
ros2 pkg create --build-type ament_python <my_package_name>

excutable 만들기
ros2 pkg create --build-type ament_cmake --node-name my_node
ros2 pkg create --build-type ament_python --node-name my_node

ws 이동 후 colcon build
```

```
특정 패키지만 빌드 : colcon build --packages-select my_package

빌드 후 ws에서 source install/setup.bash (call C:\Users\SSAFY\D
후 ros2 run <package명> <node명> 으로 노드 실행

코드 실행할 때 필요한 python 모듈 설치
pip install squaternion

ROS에서 활용하는 토픽명
/battery_state
/cmd_vel
/imu
/joint_states
/magnetic_field
/odom
/parameter_events
/robot_description
/rosout
/scan
/sensor_state
/tf
/tf_static
```

OpenCR Setup

```
[참고 링크]
https://emanual.robotis.com/docs/en/platform/turtlebot3/openc

$ sudo dpkg --add-architecture armhf
$ sudo apt update
$ sudo apt install libc6:armhf

$ export OPENCR_PORT=/dev/ttyACM0
$ export OPENCR_MODEL=burger
$ rm -rf ./opencr_update.tar.bz2

$ wget https://github.com/ROBOTIS-GIT/OpenCR-Binaries/raw/mas
```

```
$ tar -xvf ./opencr_update.tar.bz2

$ cd ~/opencr_update
$ ./update.sh $OPENCR_PORT $OPENCR_MODEL.opencr
```

Quick Start Guide

```
[기본]
ros2 launch turtlebot3_bringup robot.launch.py

[SLAM]
ros2 launch turtlebot3_cartographer cartographer.launch.py

[map save]
ros2 run nav2_map_server map_saver_cli -f ~/map_data/map

[load_map & nav2_launch]
ros2 launch turtlebot3_navigation2 navigation2.launch.py map:

# 검은색 셀은 실제 장애물
# 하늘색 영역은 로봇 중심 위치가 이 영역으로 오면 충돌하게 되는 지점
# 굵은 빨간색 픽셀은 경계선을 표시
# 보라색(파란색)은 안전한 지역을 의미

[remote_control]|
ros2 run turtlebot3_teleop teleop_keyboard

[model]
source ros_setup.sh

[PID 제어]
ros2 launch pid_controller_example pid_controller_launch.py
```