포팅매뉴얼

Index

---

1. 프로젝트 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, Mattermost, Gerrit
- 테스트 : Postman
- UI/UX : Figma

2. 프로젝트 개발 환경

Frontend

- Visual Studio Code : `1.85.1`
- React : `18.2.56`
- React-dom : `18.2.0`
- Typescript : `5.2.2`

- Node.js : `20.10.0`
- npm: `10.4.0`
- Vite : `5.1.5`
- Zustand: `4.5.2`

Backend

- IntelliJ : `2023.03`
- Java : `17`
- SpringBoot : `3.2.3`
- SpringSecurity : `3.2.3`
- JPA : `3.2.3`
- Lombok : `1.18.30`
- Python : `3.10.11`
- Kafka : `3.7.0`
- RabbitMq : `3.13.2`

DB

- MySQL : `8.3.0`
- Redis : `7.2.4`
- MongoDb : `7.0.9`
- Elasticsearch : `7.17.10`

Service

- NginX : `1.18.0`
- Jenkins : `2.451`

- Docker : `25.0.5`

Server

- Ubuntu : `20.04`

3. 외부서비스

- GitHub API

4. 빌드

Frontend

```
npm i
npm run build
```

Core

```
Gradle -> build
```

Collect

```
python main.py
```

5. 프로젝트 환경 변수

Core (Spring Boot Server)

application.yml

```
spring:
  application:
    name: core

  threads:
    virtual:
      enabled: true

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: {db_url}
    username: {db_username}
    password: {db_pw}

  jpa:
    open-in-view: false
    show-sql: false
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: true
        default_batch_fetch_size: 100

  jwt:
    secret:
      auth: {jwt_secret}
      service: {jwt_secret}
  security:
    oauth2:
      client:
        registration:
          github:
            client-id: {github_client_id}
            client-secret: {github_client_secret}
```

```
            redirect-uri: {github_redirect_uri}
            authorization-grant-type: authorization_code
            scope: "read:user,user:email"
        provider:
          github:
            authorization-uri: https://github.com/login/oauth/au
            token-uri: https://github.com/login/oauth/access_tol
            user-info-uri: https://api.github.com/user
            user-name-attribute: id
server:
  port: {port}
```

Collect

.env

```
# mongodb connection
MONGO_HOST=mongodb.example.com
MONGO_PORT=27018
MONGO_USER=${MONGO_USER}
MONGO_PASSWORD=${MONGO_PASSWORD}
MONGO_DB_NAME=omegi
MONGO_COLLECTION_NAME=error_log

# mysql database connection
DATABASE_HOST=mysql.example.com
DATABASE_PORT=3307
DATABASE_NAME=omegi
DATABASE_URL="mysql+pymysql://${DATABASE_USER}:${DATABASE_PASSW

# mysql database credentials
DATABASE_USER=${DATABASE_USER}
DATABASE_PASSWORD=${DATABASE_PASSWORD}

# kafka connection
KAFKA_HOST_1=kafka1.example.com
```

```
KAFKA_HOST_2=kafka2.example.com
KAFKA_PORT=9093
KAFKA_LOG_TOPIC=error
KAFKA_LINK_TOPIC=flow
KAFKA_GROUP_ID=omegi

# jwt
JWT_SECRET=${JWT_SECRET}

# rabbit mq
RABBITMQ_HOST=rabbitmq.example.com
RABBITMQ_PORT=5673
RABBITMQ_QUEUE=omegi_queue
RABBITMQ_USER=${RABBITMQ_USER}
RABBITMQ_PASS=${RABBITMQ_PASS}

# elasticsearch
ELASTICSEARCH_HOST=elasticsearch.example.com
ELASTICSEARCH_PORT=9201
ELASTICSEARCH_INDEX=error
ELASTICSEARCH_FLOW_INDEX=flow
ELASTICSEARCH_PASSWORD=${ELASTICSEARCH_PASSWORD}

# redis
REDIS_HOST=redis-que.example.com
REDIS_PORT=6381
REDIS_FAST_QUE=fast_queue
REDIS_SLOW_QUE=slow_queue
REDIS_FLOW_QUE=flow_queue
REDIS_FAST_INTERVAL=1
REDIS_SLOW_INTERVAL=1
```

.gitignore

```
# config
**/.env
**/application.yml
```

배포

목차

1. Docker/Jenkins 설치

1.1 Docker 설치

```
sudo apt-get -y install apt-transport-https ca-certificates c
url gnupg-agent software-properties-common | curl -fsSL http
s://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
| sudo add-apt-repository "deb [arch=amd64] https://download.
docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo apt
-get -y install docker-ce docker-ce-cli containerd.io
```

1.2 Jenkins 설치

```
docker pull jenkins/jenkins:jdk17 | docker run -d --restart a
lways --env JENKINS_OPTS=--httpPort=<포트번호> -v /etc/localtim
e:/etc/localtime:ro -e TZ=Asia/Seoul -p <포트번호>:<포트번호>-v /
jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/do
cker.sock -v /usr/local/bin/docker-compose:/usr/local/bin/doc
ker-compose --name jenkins -u root jenkins/jenkins:jdk17
```

1.3 Jenkins 내부 Docker 패키지 설치

```
apt-get update && apt-get -y install apt-transport-https ca-c
ertificates curl gnupg2 software-properties-common && curl -f
sSL https://download.docker.com/linux/$(. /etc/os-release; ec
ho "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && add-apt-r
epository "deb [arch=amd64] https://download.docker.com/linu
x/$(. /etc/os-release; echo "$ID") $(lsb_release -cs) stable"
&& apt-get update && apt-get -y install docker-ce
```

2. NginX 설정

2-1. SSL 설정

```
sudo snap install --classic certbot | sudo certbot --nginx -d
<등록할 도메인 주소>
```

2-2. 리버스 프록시 설정

1) nginx.conf

- 파일 위치 : etc/nginx/nginx.conf

```
# run nginx in foreground
daemon off;
pid /run/nginx/nginx.pid;
```

```
user npm;

# Set number of worker processes automatically based on numbe
r of CPU cores.
worker_processes auto;

# Enables the use of JIT for regular expressions to speed-up
their processing.
pcre_jit on;

error_log /data/logs/fallback_error.log warn;

# Includes files with directives to load dynamic modules.
include /etc/nginx/modules/*.conf;

events {
        include /data/nginx/custom/events[.]conf;
}

http {
        include                         /etc/nginx/mime.types;
        default_type                    application/octet-strea
m;
        sendfile                        on;
        server_tokens                   off;
        tcp_nopush                      on;
        tcp_nodelay                     on;
        client_body_temp_path           /tmp/nginx/body 1 2;
        keepalive_timeout               90s;
        proxy_connect_timeout           90s;
        proxy_send_timeout              90s;
        proxy_read_timeout              90s;
        ssl_prefer_server_ciphers       on;
        gzip                            on;
        proxy_ignore_client_abort       off;
        client_max_body_size            2000m;
```

```
        server_names_hash_bucket_size 1024;
        proxy_http_version            1.1;
        proxy_set_header              X-Forwarded-Scheme $sch
eme;
        proxy_set_header              X-Forwarded-For $proxy_
add_x_forwarded_for;
        proxy_set_header              Accept-Encoding "";
        proxy_cache                   off;
        proxy_cache_path              /var/lib/nginx/cache/pu
blic  levels=1:2 keys_zone=public-cache:30m max_size=192m;
        proxy_cache_path              /var/lib/nginx/cache/pr
ivate levels=1:2 keys_zone=private-cache:5m max_size=1024m;

        log_format proxy '[$time_local] $upstream_cache_statu
s $upstream_status $status - $request_method $scheme $host
"$request_uri" [Client $remote_addr] [Length $body_bytes_sen
t] [Gzip $gzip_ratio] [Sent-to $server] "$http_user_agent"
"$http_referer"';
        log_format standard '[$time_local] $status - $request
_method $scheme $host "$request_uri" [Client $remote_addr] [L
ength $body_bytes_sent] [Gzip $gzip_ratio] "$http_user_agent"
"$http_referer"';

        access_log /data/logs/fallback_access.log proxy;

        # Dynamically generated resolvers file
        include /etc/nginx/conf.d/include/resolvers.conf;

        # Default upstream scheme
        map $host $forward_scheme {
                default http;
        }

        # Real IP Determination

        # Local subnets:
```

```
        set_real_ip_from 10.0.0.0/8;
        set_real_ip_from 172.16.0.0/12; # Includes Docker sub
net
        set_real_ip_from 192.168.0.0/16;
        # NPM generated CDN ip ranges:
        include conf.d/include/ip_ranges.conf;
        # always put the following 2 lines after ip subnets:
        real_ip_header X-Real-IP;
        real_ip_recursive on;

        # Custom
        include /data/nginx/custom/http_top[.]conf;

        # Files generated by NPM
        include /etc/nginx/conf.d/*.conf;
        include /data/nginx/default_host/*.conf;
        include /data/nginx/proxy_host/*.conf;

}
```

2) include 된 /data/nginx/proxy_host/*.conf

```
# ------------------------------------------------------------
-
# k10a308.p.ssafy.io
# ------------------------------------------------------------
-
server {
   set $forward_scheme http;
   set $server "frontend";
   set $port 5173;

   listen 80;
   listen [::]:80;
```

```
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name k10a308.p.ssafy.io;

    # Let's Encrypt SSL
    include conf.d/include/letsencrypt-acme-challenge.conf;
    include conf.d/include/ssl-ciphers.conf;
    ssl_certificate /etc/letsencrypt/live/npm-10/fullchain.pe
m;
    ssl_certificate_key /etc/letsencrypt/live/npm-10/privkey.p
em;

    # Asset Caching
    include conf.d/include/assets.conf;

    # Block Exploits
    include conf.d/include/block-exploits.conf;

    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $http_connection;
    proxy_http_version 1.1;

    access_log /data/logs/proxy-host-1_access.log proxy;
    error_log /data/logs/proxy-host-1_error.log warn;

    location /portainer/ {
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header Connection "";
        rewrite ^/portainer(.*)$ /$1 break;
        proxy_pass http://portainer:9000;
    }

    location /portainer/api/websocket/ {
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
```

```
        proxy_http_version 1.1;
        proxy_pass http://portainer:9000/api/websocket/;
    }

    location /portainer {
        return 301 /portainer/;
    }

    location /portainer/public/ {
        proxy_pass http://portainer:9000/public/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarde
d_for;
    }

    location /api {
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Scheme $scheme;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://core:8081;

        # Asset Caching
        include conf.d/include/assets.conf;

        # Block Exploits
        include conf.d/include/block-exploits.conf;

        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $http_connection;
        proxy_http_version 1.1;
    }

    location / {
```

```
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $http_connection;
        proxy_http_version 1.1;

        # Proxy!
        include conf.d/include/proxy.conf;
    }

    # Custom
    include /data/nginx/custom/server_proxy[.]conf;
}
```

3. Kafka 설치

3.1 docker-compose.yml

```
services:
 kafdrop:
   image: obsidiandynamics/kafdrop:4.0.1
   container_name: kafdrop
   restart: "no"
   ports:
     - "19000:9000"
   environment:
     KAFKA_BROKERCONNECT: "kafka1:9092,kafka2:9092"
   depends_on:
     - "kafka1"
     - "kafka2"
   networks:
     - omegi-net

 zookeeper1:
   container_name: zookeeper1
```

```
    image: zookeeper:latest
    environment:
      ZOO_MY_ID: 1
      ZOO_SERVERS: server.1=zookeeper1:2888:3888;2181 server.2
=zookeeper2:2888:3888;2181 server.3=zookeeper3:2888:3888;2181
    networks:
      - omegi-net

  zookeeper2:
    container_name: zookeeper2
    image: zookeeper:latest
    environment:
      ZOO_MY_ID: 2
      ZOO_SERVERS: server.1=zookeeper1:2888:3888;2181 server.2
=zookeeper2:2888:3888;2181 server.3=zookeeper3:2888:3888;2181
    networks:
      - omegi-net

  zookeeper3:
    container_name: zookeeper3
    image: zookeeper:latest
    environment:
      ZOO_MY_ID: 3
      ZOO_SERVERS: server.1=zookeeper1:2888:3888;2181 server.2
=zookeeper2:2888:3888;2181 server.3=zookeeper3:2888:3888;2181
    networks:
      - omegi-net

  kafka1:
    container_name: kafka1
    hostname: kafka1
    image: bitnami/kafka:latest
    depends_on:
      - zookeeper1
      - zookeeper2
      - zookeeper3
```

```
    ports:
      - "19092:19092"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka1:9092,EXTER
NAL://k10a308.p.ssafy.io:19092
      KAFKA_LISTENERS: INTERNAL://0.0.0.0:9092,EXTERNAL://0.0.
0.0:19092
      KAFKA_ZOOKEEPER_CONNECT: zookeeper1:2181,zookeeper2:218
1,zookeeper3:2181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:PLAINTEX
T,EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 2
      KAFKA_LOG_RETENTION_HOURS: 24
      KAFKA_MESSAGE_MAX_BYTES: 1048576 # 1MB
    networks:
      - omegi-net

 kafka2:
    container_name: kafka2
    image: bitnami/kafka:latest
    depends_on:
      - zookeeper1
      - zookeeper2
      - zookeeper3
    ports:
      - "29092:29092"
    environment:
      KAFKA_BROKER_ID: 2
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://kafka2:9092,EXTER
NAL://k10a308.p.ssafy.io:29092
      KAFKA_LISTENERS: INTERNAL://0.0.0.0:9092,EXTERNAL://0.0.
0.0:29092
      KAFKA_ZOOKEEPER_CONNECT: zookeeper1:2181,zookeeper2:218
1,zookeeper3:2181
```

```
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INTERNAL:PLAINTEX
T,EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 2
      KAFKA_LOG_RETENTION_HOURS: 24
      KAFKA_MESSAGE_MAX_BYTES: 1048576
    networks:
      - omegi-net

networks:
 omegi-net:
   external: true
```

4. Elasticsearch 설치

4.1 docker-compose.yml

```
version: '3'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.1
7.10
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=true
      - ELASTIC_PASSWORD=ssafy308
      - "ES_JAVA_OPTS=-Xms2g -Xmx2g"
      - xpack.watcher.enabled=false
      - xpack.ml.enabled=false
    ports:
      - 9200:9200
    networks:
```

```
      - omegi-net
    volumes:
      - elasticsearch-data:/usr/share/elasticsearch/data

volumes:
  elasticsearch-data:

networks:
  omegi-net:
    external: true
```

5. MongoDb 설치

5.1 docker-compose.yml

```
version: '3.8'
services:
  mongodb:
    container_name: mongodb
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "27017:27017"
    volumes:
      - mongodb_data:/data/db
    networks:
      - omegi-net

networks:
  omegi-net:
    external: true
```

```
volumes:
  mongodb_data:
    driver: local
```

5.2 Dockerfile

```
FROM mongo:latest

ENV MONGO_INITDB_ROOT_USERNAME={name}
ENV MONGO_INITDB_ROOT_PASSWORD={password}
ENV MONGO_INITDB_DATABASE={db}

COPY init-mongo.js /docker-entrypoint-initdb.d/

EXPOSE 27017
```

5.3 init-mongo.js

```
db.createCollection("error_log");
```

6. MySql 설치

6.1 docker-compose.yml

```
version: '3'
services:
  mysql:
    image: mysql:latest
    container_name: mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: {db_root_pw}
```

```
      MYSQL_DATABASE: {db_name}
      MYSQL_USER: {db_user}
      MYSQL_PASSWORD: {db_ps}
    volumes:
      - ./mysql-data:/var/lib/mysql
    ports:
      - "3306:3306"
```

7. Redis 설치

7.1 docker-compose.yml

```
version: '3'

services:
  redis-que:
    container_name: redis
    image: redis
    command: redis-server /usr/local/etc/redis/redis.conf
    ports:
      - "6380:6380"
    volumes:
      - ./redis.conf:/usr/local/etc/redis/redis.conf
      - redis_data:/data
    networks:
      - omegi-net

volumes:
  redis_data:

networks:
  omegi-net:
    external: true
```

8.RabbitMq 설치

8.1 docker-compose.yml

```
version: '3'
services:
  rabbitmq:
    image: rabbitmq:3-management-alpine
    container_name: rabbitmq
    ports:
      - "5672:5672"
      - "15672:15672"
    volumes:
      - ./.docker/rabbitmq/etc/:/etc/rabbitmq/
    networks:
      - omegi-net
    environment:
      - RABBITMQ_DEFAULT_USER=omegi
      - RABBITMQ_DEFAULT_PASS=ssafy308@omegi
    restart: always

volumes:
  rabbitmq_data:

networks:
  omegi-net:
    external: true
```

9. Core 배포 (Spring Boot)

9.1 Spring Dockerfile

Dockerfile

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

FROM openjdk:21-jdk

ARG JAR_FILE=./build/libs/*.jar

ADD ${JAR_FILE} app.jar

ENTRYPOINT ["java","-Duser.timezone=Asia/Seoul", "-jar","/ap
p.jar"]
```

9.2 Jenkins 파이프라인 작성

```
tools {
    jdk 'jdk-21'
}

environment {
    JAVA_HOME = "tool jdk-21"
    imageName = '${IMAGE_NAME}'
    registryCredential = '${REGISTRY_CREDENTIAL}'
    dockerImage = ''
    releaseServerAccount = '${RELEASE_SERVER_ACCOUNT}'
    releaseServerUri = '${RELEASE_SERVER_URI}'
    releasePort = '${RELEASE_PORT}'
    gitBranch = '${GIT_BRANCH}'
    gitCredentialsId = '${GIT_CREDENTIALS_ID}'
    gitUrl = '${GIT_URL}'
}

stages {
```

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

    stage('Add Env') {
        steps {
            dir('backend/core') {
                withCredentials([file(credentialsId: '${ENV_C
REDENTIALS_ID}', variable: 'yml')]) {
                    sh 'cp ${yml} src/main/resources/applicat
ion.yml'
                }
            }
        }
    }

    stage('Jar Build') {
        steps {
            dir('backend/core') {
                sh 'chmod +x ./gradlew'
                sh './gradlew clean bootJar'
            }
        }
    }

    stage('Image Build & DockerHub Push') {
        steps {
            dir('backend/core') {
                script {
                    docker.withRegistry('', registryCredentia
l) {
                        sh "docker buildx create --use --name
```

```
mybuilder"
                        sh "docker buildx build --platform li
nux/amd64 -t $imageName:$BUILD_NUMBER --push ."
                        sh "docker buildx build --platform li
nux/amd64 -t $imageName:latest --push ."
                    }
                }
            }
        }
    }

    stage('Service Restart') {
        steps {
            sshagent(credentials: ['${SSH_CREDENTIALS_ID}'])
{
                script {
                    sh '''
                        if ssh -o StrictHostKeyChecking=no $r
eleaseServerAccount@$releaseServerUri "sudo docker ps -a --fi
lter name=core --format '{{.Names}}' | grep -q core"; then
                            ssh -o StrictHostKeyChecking=no
$releaseServerAccount@$releaseServerUri "sudo docker stop cor
e"
                            ssh -o StrictHostKeyChecking=no
$releaseServerAccount@$releaseServerUri "sudo docker rm -f co
re"
                        fi
                    '''

                    sh "ssh -o StrictHostKeyChecking=no $rele
aseServerAccount@$releaseServerUri 'sudo docker pull $imageNa
me:latest'"

                    sh '''
                        ssh -o StrictHostKeyChecking=no $rele
aseServerAccount@$releaseServerUri "sudo docker run -i -e TZ=
```

```
Asia/Seoul -e 'SPRING_PROFILES_ACTIVE=prod' --name core -p $r
eleasePort:$releasePort --network ${DOCKER_NETWORK} -d $image
Name:latest"
                    '''
                }
            }
        }
    }
}
```

10. Frontend 배포 (React)

10.1 React Dockerfile

Dockerfile

```
FROM nginx:latest

RUN mkdir /app

WORKDIR /app

RUN mkdir ./build

ADD ./dist ./build

RUN rm /etc/nginx/conf.d/default.conf

COPY ./nginx.conf /etc/nginx/conf.d

EXPOSE 5173
CMD ["nginx", "-g", "daemon off;"]
```

10.2 NginX 설정

nginx.conf

```
server {
    listen 5173;
    location / {
        root    /app/build;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

10.3 Jenkins 파이프라인 작성

```
tools {
    nodejs "nodejs"
}

environment {
    imageName = '${IMAGE_NAME}'
    registryCredential = '${REGISTRY_CREDENTIAL}'
    dockerImage = ''
    releaseServerAccount = '${RELEASE_SERVER_ACCOUNT}'
    releaseServerUri = '${RELEASE_SERVER_URI}'
    releasePort = '${RELEASE_PORT}'
    gitBranch = '${GIT_BRANCH}'
    gitCredentialsId = '${GIT_CREDENTIALS_ID}'
    gitUrl = '${GIT_URL}'
}

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

```
        }
    }

    stage('Node Build') {
        steps {
            dir('frontend') {
                sh 'npm install'
                sh 'npm run build'
            }
        }
    }

    stage('Image Build & DockerHub Push') {
        steps {
            dir('frontend') {
                script {
                    docker.withRegistry('', registryCredentia
l) {
                        sh "docker buildx create --use --name
mybuilder"
                        sh "docker buildx build --platform li
nux/amd64 -t $imageName:$BUILD_NUMBER --push ."
                        sh "docker buildx build --platform li
nux/amd64 -t $imageName:latest --push ."
                    }
                }
            }
        }
    }

    stage('Before Service Stop') {
        steps {
            sshagent(credentials: ['${SSH_CREDENTIALS_ID}'])
{
                sh '''
                    if test "`ssh -o StrictHostKeyChecking=no
```

```
$releaseServerAccount@$releaseServerUri "docker ps -aq --filt
er ancestor=$imageName:latest"`"; then
                        ssh -o StrictHostKeyChecking=no $rele
aseServerAccount@$releaseServerUri "docker stop $(docker ps -
aq --filter ancestor=$imageName:latest)"
                        ssh -o StrictHostKeyChecking=no $rele
aseServerAccount@$releaseServerUri "docker rm -f $(docker ps
-aq --filter ancestor=$imageName:latest)"
                        ssh -o StrictHostKeyChecking=no $rele
aseServerAccount@$releaseServerUri "docker rmi $imageName:lat
est"
                    fi
                '''
            }
        }
    }

    stage('DockerHub Pull') {
        steps {
            sshagent(credentials: ['${SSH_CREDENTIALS_ID}'])
{
                sh "ssh -o StrictHostKeyChecking=no $releaseS
erverAccount@$releaseServerUri 'sudo docker pull $imageName:l
atest'"
            }
        }
    }

    stage('Service Start') {
        steps {
            sshagent(credentials: ['${SSH_CREDENTIALS_ID}'])
{
                sh '''
                    ssh -o StrictHostKeyChecking=no $releaseS
erverAccount@$releaseServerUri "sudo docker run -i -e TZ=Asi
a/Seoul --name frontend --network ${DOCKER_NETWORK} -p $relea
```

```
sePort:$releasePort -d $imageName:latest"
                '''
            }
        }
    }
}
```

11. Collect 배포 (Python)

11-1. Python Dockerfile 작성

Dockerfile

```
FROM python:3.10

WORKDIR /app

COPY requirements.txt ./

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

COPY . .

CMD ["python", "./main.py"]
```

11-2. docker-compose.yml

```
version: '3.8'
services:
  comsumer:
    build: .
    image: collect-server:latest
    container_name: collect-server
    networks:
```

```
      - omegi-net
    ports:
      - "8051:8051"

networks:
  omegi-net:
    external: true
```

11-3. bash

```
git clone {url}
cd {dir}/collect
docker-compose build
docker-compose up -d
```

외부 서비스 이용

Github 로그인 API

1) 깃허브 설정 → Develop settings → New Github App

2) Homepage url, Callback url 설정 후 사용